DTIC FILE COPY

E 801 780

(2)

AFATL-TR-88-117, VOL II

Program EAGLE User's Manual

Vol II-Surface Generation Code

AD-A204 142

Joe F Thompson
Boyd Gatlin

DEPARTMENT OF AEROSPACE ENGINEERING
MISSISSIPPI STATE UNIVERSITY
DRAWER A
MISSISSIPPI STATE, MS 39762

DTIC
ELECTE
OCT 1 2 1988
S D
H

SEPTEMBER 1988

INTERIM REPORT FOR PERIOD OCTOBER 1986-SEPTEMBER 1988

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

AIR FORCE ARMAMENT LABORATORY
Air Force Systems Command ■ United States Air Force ■ Eglin Air Force Base, Florida

88 1011 225

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility nor any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Public Affairs Office has reviewed this report, and it is releasable to the National Technical Information Service (NTIS), where it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

[illegible]

STEPHEN C. KORN
Technical Director, Aeromechanics Division

Please do not request copies of this report from the Air Force Armament Laboratory. Copies may be obtained from DTIC. Address your request for additional copies to:

Defense Technical Information Center
Cameron Station
Alexandria, VA 22304-6145

If your address has changed, if you wish to be removed from our mailing list, or if your organization no longer employs the addressee, please notify AFATL/FXA , Eglin AFB FL 32542-5434, to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release, distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFATL-TR-88-117, VOL II
6a. NAME OF PERFORMING ORGANIZATION Dept of Aerospace Engineering Mississippi State University 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Aerodynamics Branch Aeromechanics Division
6c. ADDRESS (City, State, and ZIP Code) Drawer A Mississippi State University MS 39762	7b. ADDRESS (City, State, and ZIP Code) Air Force Armament Laboratory Eglin Air Force Base FL 32542-5434
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Aerodynamics Brn Aeromechanics Division 8b. OFFICE SYMBOL (If applicable) AFATL/FXA	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F08635-84-C-0228
8c. ADDRESS (City, State, and ZIP Code) Air Force Armament Laboratory Eglin Air Force Base FL 32542-5434	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 62602F; PROJECT NO. 2567; TASK NO 03; WORK UNIT ACCESSION NO. 08

11. TITLE (Include Security Classification)
Program EAGLE User's Manual, Volume II: Surface Generation Code

12. PERSONAL AUTHOR(S)
Joe E. Thompson, Boyd Gatlin

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Interim	FROM Oct 86 TO Sep 88	September 1988	268

16. SUPPLEMENTARY NOTATION This Volume is a joint in-house and contractor effort. Therefore, it is in contractor format.
Availability of report is specified on verso of front cover.

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
01	01	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Numerical Grid Generation
Surface Generation
Boundary - Conformal

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report provides both a user's manual and a detailed description of a general surface code for the construction of boundary surface input to a grid generation code. This code generates various forms of curved surfaces and performs the manipulation of such surfaces necessary for assembly into a composite structure for boundaries of general three-dimensional regions.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED [✓ SAME AS RPT.] ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL John R. Cipolla	22b. TELEPHONE (Include Area Code) (904) 882-3124 22c. OFFICE SYMBOL AFATL/FXA

DD Form 1473, JUN 86 Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

SUMMARY

This volume serves both as a user's manual (Part I) and a detailed documentation of the operation (Part II) for the boundary code, which generates boundary surfaces for input to the grid code (Volume III). This boundary code can generate surfaces or distribute points on input surfaces. It can also perform various geometric manipulations, such as transformations and scaling, on surfaces.

DTIC COPY INSPECTED

Accession For
NTIS GRA&I [✓]
DTIC TAB ☐
Unannounced ☐
Justification

By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

PREFACE

Volume II of a four-volume set documents the usage of Program EAGLE (Eglin Arbitrary Geometry ImpLicit Euler) to generate the surfaces describing the computational region. The surface generation system serves as a front-end to the grid generation system (Volume III), generating surfaces (curves) to be input to the grid system as segments of the boundary region within which the grid is to be constructed.

This report was prepared by Drs Joe F. Thompson and Boyd Gatlin of Mississippi State University (MSU), Starkville, MS. The work was performed under Work Unit 25670308 from 1 October 1986 to 30 September 1988.

The principal investigator of the surface and grid generation theory has been Dr Joe F. Thompson of MSU. The principal investigators of the flow code theory have been Dr David L. Whitfield of MSU, and Dr Dave M. Belk and Mr L. Bruce Simpson of the Computational Fluid Dynamics Section (AFATL/FXA). Capt Jon S. Mounts of the Computational Fluid Dynamics Section (AFATL/FXA) has increased the utility of the flow code through user-oriented inputs and outputs, extensive error checking, and calculation of component forces and moments.

The program manager for the development of Program EAGLE has been Dr Lawrence E. Lijewski of the Computational Fluid Dynamics Section.

ACKNOWLEDGEMENT

This project has involved cooperative efforts between the contractor and the sponsor, and Dr Lawrence Lijewski of Eglin Air Force Base has contributed significantly both through direction and involvement. Codes of this size evolve as usage uncovers problems and new requirements. In this regard, Lt Agusto Martinez has been instrumental, and was also heavily involved in the incorporation of the use of the solid state disk. Dr Dave Belk, Mr L. Bruce Simpson, Mr Yen Tu, Mr John Cipolla, Mr Lynn Lewis, Capt Jon Mounts, and Lt Montgomery Hughson of Eglin have also contributed. Graduate students Yeon Seok Chae, Col Hyun Jun Kim, and Maj Yong Hyun Yoon of Mississippi State University have contributed significantly to the checkout of these codes. Mr Chae did the artwork. Finally, Mrs Susan Triplett Price of Mississippi State did all the typing with competence, dedication, and extra hours.

This effort was conducted during the period 1 October 1986 to 30 September 1988 for the Air Force Armament Laboratory, Eglin Air Force Base, FL 32542-5434, under Contract F08635-84-C-0228.

TABLE OF CONTENTS

PART I - USER'S MANUAL

A. INTRODUCTION. 7

Operations List 9

B. COMMON FEATURES 10

1. Adjustable Dimension Parameters 10
2. Placing a Surface(Curve) on Core or File Storage. 11
3. Retrieving a Surface(Curve) from Core or File Storage . . . 13
4. Reading a Surface(Curve). 15
5. Printing and/or Plotting a Surface(Curve) 16
6. Setting a Point Distribution with Specified Spacings. . . . 18
7. Use of Internally Calculated Values 21
8. Curves on Curved Surfaces 22
9. Numbered Points 23
10. Spacing from Existing Curve 23
11. Scratch Files 24

C. GENERATION OF CURVES. 25

1. Plane Conic-Section Curves. 25
2. Cubic Space Curve Between Two Points. 29
3. Cubic Space Curve Connecting Two Surfaces(Curves) 31
4. Straight Line 34
5. Generation of a Curve as the Intersection of Two Surfaces . 36

D. GENERATION OF SURFACES. 37

1. Flat Conic-Section Surfaces 37
2. Conic-Section Surfaces. 39
3. Cubic Patch Surface 45
4. Generation of a Surface by Rotation of a Curve. 48
5. Generation of a Surface by Stacking Curves. 52
6. Generation of a Surface by Blending Curves. 55
7. Generation of a Surface by Transfinite Interpolation. . . . 58
8. Generation of a Tensor-Product Surface (Coon's Patch) . . . 59
9. Generation of a Surface from a Surface-Parametric Coordinate Lattice. 60

E. AUXILIARY OPERATIONS. 62

1. Distributing Points on a Curve with Specified Spacing . . . 62
2. Distributing Points on a Curve According to Curvature . . . 63
3. Scaling a Surface(Curve). 63

4. Transforming a Surface(Curve) 64
5. Designating a Curve to be Rotated, Stacked, or Blended. . . 67
6. Designating a Curve as the Axis Curve of a Stack. 67
7. Designating a Curve as an Edge Curve for a Transfinite Interpolation Surface 68
8. Setting a Corner for a Tensor-Product Surface 69
9. Construction of a General Tab Curve 70
10. Construction of an Edge Tab Curve 72
11. Construction of a Tab Point 74
12. Direct Setting of a Tab Point 76
13. Designation of an End Point 77
14. Designation of an End Point and Slope Vector. 78
15. Splining a Surface(Curve) 78
16. Placing a Surface(Curve) in Current Position. 79
17. Storing a Surface(Curve). 79
18. Reversing Point Progression and/or Surface Dimensions . . . 79
19. Concatenation of Surfaces(Curves) 81
20. Duplication of a Surface(Curve) Segment 82
21. Combining Cores and/or Files onto One Core or File. 83
22. Copying Cores or Files. 85
23. Setting an Input Value as a Sum or Product. 85
24. Projection on a Plane 86
25. Cartesian Coordinates at Numbered Points. 87
26. Surface Parametric Coordinates of a Surface(Curve). 87
27. Distributing Points on a Surface with Specified Spacing . . 88

F. OTHER OPERATIONS. 89

1. Relative Distribution 89
2. Arc Length on a Curve 89
3. Unit Tangent and Principal Normal, Curvature, and Arc Length on a Curve 90
4. Unit Tangents and Normal on a Surface 90

G. ERROR MESSAGES. 92

PART II - CODE OPERATION

A. PARAMETERS AND VARIABLES. 115

1. Primary Arrays. 115
2. Surface(Curve) Parameters 116
3. I/O Parameters. 118
4. Distribution Parameters 119
5. Spline Parameters 120

B. INPUT AND SETUP 121

C. RESPONSES TO NAMELIST/INPUT 124

1. Storing, printing, and plotting a surface 124
2. Retrieval of a surface. 125
3. Relative distributions. 126
4. Surface(Curve) Spline 127

5. ITEM="CURDIST". 127
6. ITEM="DISTRIB". 128
7. ITEM="CURVAT" 128
8. ITEM="ARC". 129
9. ITEM="CURVEC" 129
10. ITEM="SURVEC" 129
11. ITEM="GETEND" 130
12. ITEM="GENTAB" 130
13. ITEM="EDGETAB". 131
14. ITEM="CURTAB" 131
15. ITEM="SETTAB". 132
16. ITEM="SETCOR". 132
17. ITEM="SCALE" 133
18. ITEM="TRANS" 133
19. ITEM="CONICUR" 134
20. ITEM="SCURVE". 134
21. ITEM="FLATSUR" 136
22. ITEM="CONISUR" 137
23. ITEM="SPLINE". 138
24. ITEM="INSEC" 138
25. ITEM="EDGECUR" 138
26. ITEM="AXIS". 139
27. ITEM="BOUNCUR" 139
28. ITEM="ROTATE". 140
29. ITEM="STACK" 142
30. ITEM="BLEND" 143
31. ITEM="TRANSUR" 144
32. ITEM="TENSUR". 145
33. ITEM="PARCOR". 146
34. ITEM="PATCH" 147
35. ITEM="LINE". 148
36. ITEM="CURRENT" 149
37. ITEM="SWITCH". 149
38. ITEM="OUTPUT". 149
39. ITEM="INSERT". 150
40. ITEM="EXTRACT" 150
41. ITEM="COMBINE" 151
42. ITEM="COPY". 152
43. ITEM="POINT" 152
44. ITEM="CORPAR". 153
45. ITEM="SETVAL". 153
46. ITEM="SETNUM". 153
47. ITEM="SURDIST" 154

D. SUBROUTINES

Subroutines List 155

1. CURDIST. 157
2. RELDIST. 157
3. SING,COSG,TANG,ACOSG 159
4. AITKEN 159
5. ARCLNGT. 160
6. CURVEC 160

7. SURVEC 162
8. GENTAB 164
9. SCAL 167
10. TRANS. 167
11. CONICUR. 168
12. DET3 172
13. DET4 172
14. FLATSUR. 172
15. CONISUR. 173
16. INTSEC 178
17. ROTATE 181
18. STACK. 186
19. BLEND. 190
20. PATCH. 191
21. EDGETAB. 194
22. INTERP1,INTERP2, and INTERP3 197
23. LINE 197
24. PUT. 198

Core Storage 198
File Storage 200
Print. 201
Plot 201

25. GET. 202

From core. 203
From file. 203
From namelist. 205
From the current array 205

26. SWITCH 205
27. INSERT 206
28. EXTRACT. 207
29. SCURVE 208
30. CURTAB 209
31. FACDIST. 211
32. SPLINE 212
33. SPARC. 213
34. TRANSUR. 214
35. TENSUR 216
36. CORPAR 218
37. TANPAR 222
38. PARCOR 224
39. SPLNSUR. 225
40. SURSPL 225
41. CURSPL 226

APPENDIX A

DISTRIBUTION FUNCTIONS 227

APPENDIX B

CONIC-SECTION CURVES 234

1. Circle. 234
2. Ellipse 234
3. Circular arc. 235
4. Elliptical arc. 236
5. Parabola. 237
6. Hyperbola 238

APPENDIX C

CONIC-SECTION SURFACES 240

1. Sphere. 240
2. Ellipsoid 240
3. Spherical segment 241
4. Ellipsoidal segment 244
5. Elliptic cone 244
6. Elliptic paraboloid 245

APPENDIX D

CUBIC SPACE CURVE. 248

APPENDIX E

RADIUS OF CURVATURE. 249

APPENDIX F

SURFACE(CURVE) SPLINE. 250

APPENDIX G

SPLINE 252

APPENDIX H

NEWTON ITERATION FOR INTERSECTION. 254

REFERENCES 257

INTRODUCTION

The surface generation system serves as a front-end to the grid generation system (Vol. III), generating surfaces(curves) to be input to the grid system as segments of the boundary of the region within which the grid is to be constructed. The surface system can generate certain generic surfaces, or can receive general surfaces from the grid system or other source as input for further processing. Curves can also be rotated, stacked or blended to form surfaces. In any case, the surfaces can be scaled, transformed and concatenated to form general boundary segments. This system also can generate curvilinear grids on curved surfaces in terms of surface parametric coordinates by interpolation on the splined surface and can generate surface intersections.

The code operates by responding to a series of commands given through NAMELIST input. Each read of the NAMELIST invokes a single operation, e.g. the generation of a generic surface or the transformation of a surface. General boundary segments are constructed by a sequence of these operations, culminating in storage of the surface for later input to the grid generation system. The code can also plot the surface at any stage of development.

In the following sections the use of the code is described first in Part I, and a general discussion of the operation of each subroutine is given thereafter in Part II and the Appendices. This latter discussion is not essential to the use of the code, but is provided to allow for greater understanding and possible modification of the code.

NEW FEATURES

The 1988 version of the front-end boundary code for the EAGLE grid code contains several modifications and improvements of certain features of the original code. The major thrust of these new features is to simplify the construction of complicated boundary configurations. Reference may be made to a new section entitled Easy EAGLE (Section 4 of Volume I) in which the general use of the code is discussed. This new section is intended to provide the user with essential information on the most commonly used features of the code. This section should hopefully allow the new user to quickly get into operation.

The construction of complicated boundary surfaces has been greatly simplified by making provision for the addressing of points by point numbers instead of by the three Cartesian coordinates (Section I-B9). Boundary segments and spacings can also be addressed by numbers. Spacings and the number of points on segments can be defined in one place and then used on different segments having common values. This greatly facilitates the changing of such parameters since common values do not have to be changed explicitly throughout the runstream. These point and segment numbers can be carried from the front-end boundary code into the grid code, and this allows the number of points on the various boundary segments to be changed in the input to the front-end code without requiring changes in the input to the grid code. This feature also makes it much easier to construct the block structure from sketches of the configuration.

The most significant other new features of the front-end code are the following:

1. The input has been simplified by the use of a number of additional defaults.
2. Provision has been made for creation of a file of segments for plotting.
3. Several features regarding point distributions have been added to make it easier to set spacings to match those on other segments.
4. Provision has been made for the generation of surface parametric coordinates for use as input to the grid code for the generation of a 2D grid on a curved surface.
5. Point distributions can now be placed on surfaces.

Several bugs in the original version have also been fixed, of course. Some new sections have also been added, following the original sections. Finally, alphabetical lists of the ITEM operations and the subroutines have been added.

The following significant changes to specific operations should be noted (Details are given in the sections indicated):

<u>Section I-B2, B3</u>

The default for TRIAD has been changed to "YES".

<u>Section I-B5</u>

It is also possible to place several surfaces(curves) on a file for later plotting by using ITEM="COMBINE" (Section I-E21).

Section I-B6

If a single value is given for SPACE (or its equivalenced quantities) DISTYP and its equivalenced quantities default to "TANH". If two values are given for SPACE, DISTYP defaults to "BOTH".

The value for SPACE will be set to a value previously stored by the operation "SETVAL" (Section I-E23) if a negative integer is given for SPACE.

Section I-B9

Points can be addressed by numbers if the operation ITEM="POINT" (Section I-E25) has been used to set the Cartesian coordinates of the point. In that case a single positive integer given for R1 or R2 will be taken to be the point number, and the corresponding coordinates will be retrieved from storage.

Section I-B10

The spacing given in SPACE (or its equivalenced quantities) can be taken from that on an end of an existing curve by giving a positive integer for SPACE with RELATIV omitted.

Section I-C2, C4

A single positive integer for R1 or R2 is taken as a point number, set by a previous usage of ITEM="POINT" (Section I-B9).

Section I-D4

The inclusion of NORCOS is no longer necessary.

Section I-E1

The spacing can be set in several ways as discussed in Section I-B6.

Section I-C11, C13

If POINT="FIRST", the first point on the existing curve will be taken (same effect as POINT=1), and if POINT="LAST" the last point will be taken.

Section I-E15

If the surface was the intersected surface of the intersection operation, ITEM="INTSEC", and no later surface has been splined, it is not necessary to invoke ITEM="SPLINE" for that surface.

Section I-E18

The default for REORDER is ("SWITCH", 0,0), simply interchanging the faster and slower running directions.

Section I-E19

With curves, START is defaulted to the end of the curve, and can thus be omitted if one curve is to be added to the end of another.

Section I-E21

The operation ITEM="COMBINE" will store a table of contents of the combined file if CONTENT="YES". This table consists of the COREOUT number and the surface(curve) dimensions for each segment on the file.

This table can be read by the grid code and used there to set up the boundary configuration (cf. Section I-C23 of Volume III).

Section I-E23

TYPE has been changed to MATH.

The operation "SETVAL" now sets only real values.

The operation "SETNUM" sets integer values.

Section I-E25

The operation

ITEM="POINT"

sets the Cartesian coordinates, given in R(3), of a numbered point indicated by the positive integer given for POINT (cf. Section I-B9).

Section I-E26

The operation

ITEM="CORPAR"

generates the values of the surface parametric coordinates for each point on an existing surface(curve) constructed on a curved surface.

Section I-E27

The operation

"SURDIST"

distributes points on a surface with specified spacing at each edge.

PART I - USER'S MANUAL

A. INTRODUCTION

The various operations for the generation of surfaces(curves), or the manipulation thereof, are described below. Each of these operations is invoked by a NAMELIST input statement of the form

```
E$INPUT ITEM = "operation", quantity=value, --- $
```

where the name in quotes designates the particular operation, and values for the quantities relevant to this operation are specified following this designation, each specification being separated from the next by a comma. In these specifications, 'quantity' is the name of the quantity, and 'value' is its value, e.g. RADIUS=2.3.

Arrays appear as

```
quantity = value, value, ---
```

and repeated values in arrays can be indicated by N*value, where N is the number of values. No distinction is made on the left side of the equal sign on the input statement between quantities that are scalars or arrays unless only a value other than the first is to be given for an array, in which case the notation is

```
quantity(N) = value
```

where N indicates the position in the array. Only those entries in an array that are relevant to the operation invoked need be given. Thus an array dimensioned for two values, e.g. the number of points in each direction on a surface, is given only one value for a curve, i.e.

```
quantity = value
```

Values are given as integers or floating point numbers as appropriate to the particular quantity. Floating point numbers may be in either decimal or exponential form, e.g. 102.3 or 1.023E2. The decimal can be omitted if the value given for a floating point quantity happens to be an integer, i.e., 10 serves as well as 10.0. Only relevant quantities need be included on the input statement. All quantities are defaulted after each read, and the code checks the input for unreasonable or omitted values of relevant quantities.

In all the following discussions, capital letters on the input statement are to appear on the input exactly as given, whether on the left of an equal sign or within quotes. (The quotes also appear.) Numerical values to be given are identified by small letters. Only those quantities relevant to the particular operation involved need be included. The only essential space is that between INPUT and ITEM. Spaces around the commas and equal signs may be used for clarity. The input is terminated by the statement

```
E$INPUT ITEM = "END" $
```

In many cases a series of operations is performed to generate a final surface. When a succeeding operation is performed on the surface resulting from the immediately preceding operation, it is not necessary to store the intermediate surface since the result of each operation remains in 'current' position to be treated by the next operation. If, however, other operations intervene, or if a surface is to be used in more than one later operation, then the surface must be stored for later use.

OPERATIONS LIST

Operation	Page
ARC	129
AXIS	139
BLEND	143
BOUNCUR	139
COMBINE	151
CONICUR	134
CONISUR	137
CORPAR	153
COPY	152
CURDIST	127
CURRENT	149
CURTAB	131
CURVAT	128
CURVEC	129
DISTRIB	128
EDGECUR	138
EDGETAB	131
EXTRACT	150
FLATSUR	136
GENTAB	130
GETEND	130
INSEC	138
INSERT	150
LINE	148
OUTPUT	149
PARCOR	146
PATCH	147
POINT	152
ROTATE	140
SCALE	133
SCURVE	134
SETCOR	132
SETNUM	153
SETTAB	132
SETVAL	153
SPLINE	138
STACK	142
SURDIST	154
SURVEC	129
SWITCH	149
TENSUR	145
TRANS	133
TRANSUR	144

B. COMMON FEATURES

There are several features - storage and retrieval, printing and plotting, and the setting of point distributions - that can be done in connection with most surface(curve) generation operations by including certain quantities on the input statement for the operation. These common features are described below, and then are only referenced later in regard to the particular operations in which they are involved.

1. Adjustable Dimension Parameters

There are several dimension parameters that are set by identical PARAMETER statements in the main program and in the subroutines. These parameters can be changed by global edits.

DIM1,DIM2 - dimensions of largest surface that can be treated. (DIM1 is dimension of largest curve)

DIMSS - maximum number of points that can be stored in core. (Core storage is in an array dimensioned DIMSS. More than DIMSS points can actually be stored, in which case this array is written to file and the core space is then reused. There is no limit to the total number of points that can be stored in this manner since each version of the array is written to file as it is filled. In the interest of speed, DIMSS should be made as large as is practical to avoid excessive file I/O.) (Section I-B2)

DCOR - maximum number of surfaces(curves) that can be stored in core. (This should not normally have to be given any consideration.) (Section I-B2)

DFIL - maximum number of surfaces(curves) that can be stored on file. (This should not normally have to be given any consideration.) If there is a system limit on file numbers, DFIL should be set to 10 less than that limit. (Section I-B2)

DIMV - maximum number of points that can be read for a surface(curve) from the namelist. (Section I-B4)

DVAL - maximum number of values that can be stored for input as values of quantities on later input statements. (Section I-B7)

NVALMX - maximum number of terms that can be involved in the calculation of a stored value. (Section I-B7)

DPNT - maximum number of numbered points that can be used. (Section I-B9)

2. Placing a Surface(Curve) on Core or File Storage

Surfaces(curves) can be stored in core and/or on file. This is done by including one of the following on the input statement for the operation which generates the surface(curve):

COREOUT = core number

or

FILEOUT = file number

These numbers are positive integers, except as noted below. Only one surface(curve) can be stored on core with each storage number, and only one can be stored on file with each number. The same number can, however, be used for a core number and for a file number, since core storage and file storage are completely independent, neither implying the other.

The total number of surfaces(curves) that can be stored in core is DCOR, and the total that can be stored on file is DFIL. These limits are set by PARAMETER statements which can be changed by global edits. Error messages and termination result if these limits are exceeded.

There will also usually be system limits on the largest file number, and certain file numbers may be reserved for system use. The file numbers used in the WRITE statements are actually FILEOUT+10, so that the oft-used system files with numbers below 10 are automatically avoided. This addition of 10 to the number given by FILEOUT must be borne in mind in job control statements that get or preserve the file, e.g. the file created with FILEOUT=1 must be gotten or preserved by a job control statement referring to file 11.

Both core numbers and file numbers can be reused when overwriting is intended, and rewinding of files is automatic unless inhibited by including

REWOUT = "NO"

Rewinding is irrelevant to core storage, of course. Although core storage numbers can be reused, the storage is not released so that there is really no incentive to do so.

Files are normally written one Cartesian coordinate to a line. The format is set by FORM as follows:

"UNFORM"	:	unformatted (default)
"E"	:	E20.8
"LIST"	:	list-directed

The grid code uses the unformatted form by default also, and this form should be used throughout unless the file is intended for later use not allowing the unformatted form. If TRIAD="YES" is included on the input statement, all three coordinates will be written on one line.

It is possible to have a label printed in association with the storage operation by including

LABOUT = "___"

where a string of up to eight alphanumeric characters appears inside the quotes. This label is not placed on the storage, however, but is simply printed and thus cannot be used by the code to locate the element.

Certain operations can be done on more than one surface(curve) at the same time. In that case, the surfaces involved can be stored by including one of the following:

COREOUT= ___,___,___,---,___

or

FILEOUT= ___,___,___,---,___

where now the multiple entries are the core and/or file numbers, which do not have to be given in any order. Negative entries imply all numbers from the preceding entry (which must be positive) to the magnitude of the negative entry, e.g. 5,-9 refers to files 5,6,7,8, and 9. The single value, "SAME", given instead of storage numbers, implies all of the same storage locations retrieved on a COREIN or FILEIN for the same operation.

3. Retrieving a Surface(Curve) from Core or File Storage

This is done by including one of the following on the input statement for the operation which is to use the surface(curve):

COREIN = core number

or

FILEIN = file number

where the numbers are positive integers except as indicated below. Rewinding of files is automatic unless inhibited by including

REWIN = "NO"

Rewinding is not relevant to core storage, of course.

Such files could have been written by this code during the current run, or may have been preserved from a previous run, or may be from some other source. The surface could have been written as triads of three Cartesian coordinates per line, in a format indicated by FORM as in Section I-B2, or with each coordinate on a separate line if *TRIAD="NO"* is included.

When files are retrieved and placed in the same operation, a different form of the input file can be specified by including ITRIAD and IFORM, with the same usage explained for TRIAD and FORM. The defaults are the same as for TRIAD and FORM. If ITRIAD and IFORM are not included, the specification set by TRIAD and FORM is in effect.

A file written by the grid code (Vol. *III*, *Section* C-17) can be read by giving the negative of the file number for FILEIN. Such a file must contain only a two-dimensional surface(curve). If this surface is composed of multiple blocks, and storage is called for, each block will be stored on successive core or file numbers, starting with the one given by COREOUT or FILEOUT, e.g. a file containing three blocks read from the grid code will be stored here on files FILEOUT, FILEOUT+1, FILEOUT+2, etc. Consideration must be given to this, else some files may be overwritten inadvertently.

It is possible to have a label printed in association with the retrieval operation by including

LABIN = "___"

where a string of up to eight alphanumeric characters appears inside the quotes. This label is not used by the code.

Certain operations can be done on more than one surface(curve) at the same time. In that case, the surfaces involved can be retrieved from storage by including one of the following:

```
COREIN = ___,___,___,---,___
               or
FILEIN = ___,___,___,---,___
```

where now the multiple entries are the core and/or file numbers, which do not have to be given in any order. Negative entries imply all numbers from the preceding entry (which must be positive) to the magnitude of the negative entry, e.g, 5,-9 refers to files 5,6,7,8, and 9.

4. Reading a Surface(Curve) from the Input

A surface(curve) can be read directly on the input statement for an operation by including:

```
VALUES = ___,___,___,
         ___,___,___,
               .
               .
               .
         ___,___,___$
```

where the values given are the three Cartesian coordinates of each successive point on the surface. The succession is first along the curvilinear coordinate corresponding to the first index on the surface, and then on to successive values of the other index. The maximum number of points that can be read on a surface is DIMV, which is set by a PARAMETER statement and can be changed by a global edit.

5. Printing and/or Plotting a Surface(Curve)

A surface(curve) can be printed by including

```
OUT = "PRINT"
```

on the input statement for the generating operation. Similarly, plotting is done by including

```
OUT = "PLOT"
```

Both can be done by including

```
OUT = "PRINT", "PLOT"
```

and the order here is immaterial.

A surface that has already been generated and stored can be printed and/or plotted in the same manner using the operation ITEM="OUTPUT" (Section I-E17). (For job control statements, the plot file is 8.) There are several other plotting parameters that can be specified, though all are defaulted and therefore may be omitted.

Symbols can be put on the grid points by including

```
SYMBOL = 1 or -1
```

The former (1) gives symbols and lines, while the latter (-1) gives symbols but no lines.

The size of the plot on the screen can be set by including

```
SIZE = width, height
```

using real numbers representing inches on the screen. The default is 8" wide and 8" high.

The plots for each successive operation are normally added to the frame of the preceding plot to form a composite plot. A new frame can be called for by including

FRAME = "NEW"

All of the following parameters are irrelevant except in this case. Limits on the region of space that is plotted can be set by including

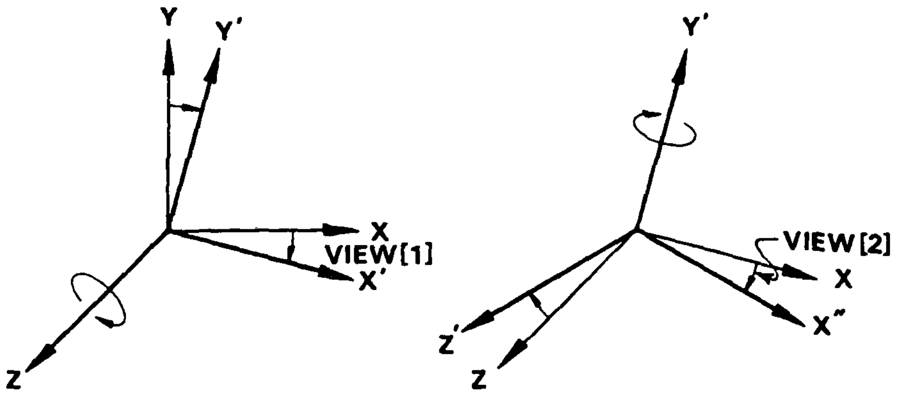
RMIN = lower x, lower y, lower z

RMAX = upper x, upper y, upper z

where these limits are real numbers. The default is to plot the entire field. The orientation of the view can be set by including

```
VIEW = angle about z-axis in degrees,
       angle about new y-axis in degrees,
       and distance from the center of the region.
```

The first angle ranges from -180 to 180, and the second from 0 to 180.



All three entries are real. The distance can be omitted if only the angles are to be set. The default is 90° for both angles, i.e., x to the right and y up, and 1000 times the maximum diagonal of the field for the distance.

It is also possible to place several surfaces(curves) on a file for later plotting by using ITEM="COMBINE" (Section I-E21). Here the form of the data can be determined by including TRIAD and FORM (Section I-B2), and the inclusion of HEAD="YES" will place a counter and the two dimensions of the surface(curve) on a single line preceding the data for each. For example, with TRIAD="YES", FORM="E" or "LIST", and HEAD="YES", the file will have the form

```
      1    N1   N2   \
      x    y    z     |
      x    y    z     }   first surface on file
           .          |
           .          |
           .         /
  ------------------
      2    N1   N2   \
      x    y    z     |
      x    y    z     }   second surface on file
           .          |
           .          |
           .         /
  ------------------
          etc.
```

where the surface dimensions are N1 x N2.

6. Setting a Point Distribution with Specified Spacings

Most of the surface(curve) generation operations allow the specification of point spacings at one or both ends of a grid line, or at a specified interior point on the line. This is done by including the following, or one of its equivalenced variations as noted in the individual operations discussed below, on the input statement for the generating operation:

DISTYP = "type"

where the type is one of the following:

BOTH	- for specified spacing at both ends
TANH or SINH	- for specified spacing at one end
INTERIOR	- for specified spacing at an interior point

(With specified spacing at one end, TANH gives the smoothest overall distribution, while SINH gives the most uniform spacing near the end. The former is more generally appropriate.) Specification at one end means at the first end. No provision is made for specification at only the second end. No generality is lost thereby because the end where the spacing is to be specified can always be made the first end, by using ITEM = "SWITCH" (Section I-E18).

The specification of the spacing at both ends is done by including

SPACE = first end spacing, second end spacing

while specification at one end is done by

SPACE = first end spacing

If the specification is at an interior point, the following are given:

SPACE = spacing

ARCINT = arc length location of interior point

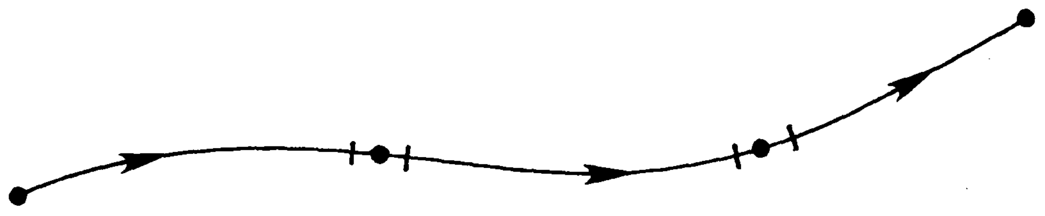
The spacings (and the arc length) in these specifications may be relative, i.e., fractions on the range 0-1, unless

RELATIV = "NO" and TOTARC = total arc length are included. There are two entries in RELATIV, corresponding to the two in SPACE, when the spacing is specified on both ends. Similarly the second entry corresponds to the arc length for specified interior spacing.

If a single value is given for SPACE (or its equivalenced quantities) DISTYP and its equivalenced quantities default to "TANH". If two values are gvien for SPACE, DISTYP defaults to "BOTH".

The value for SPACE will be set to a value previously stored by the operation "SETVAL" (Section I-E23) if a negative integer is given for SPACE. In this case the magnitude is the storage location.

If a positive integer is given for SPACE, and RELATIV is not included, the absolute spacing will be taken from that at an end of an existing curve (Section I-B10). In this case, the particular end must be indicated by including END (having two entries, corresponding to those of SPACE). The possible values for END are "FIRST" and "LAST", indicating that the spacing is to be taken from the first or last end of the existing curve. The default for END is "LAST", "FIRST" corresponding to the common case illustrated below.



The existing curve does not, however, have to be physically adjacent to an end of the curve being generated, but such will often be the case. This mode is allowed only for operations that allow absolute spacing to be specified.

It is not necessary to use the same mode for each of the two entries of SPACE.

For distributions in both directions on a surface, a terminology analogous to that given above is employed. This is described for each type of surface as they are discussed below. The following equivalenced variables are used for the distribution parameters in different operations:

DISTYP	SPACE	ARCINT	
DISTANG	SPACANG	ARCIANG	: angular distribution
DISTLON	SPACLON	ARCILON	: longitude distribution
DISTLAT	SPACLAT	ARCILAT	: latitude distribution
DISTRAD	SPACRAD	ARCIRAD	: radial distribution
DISTCUR	SPACCUR	ARCICUR	: curve distribution

7. Use of Internally Calculated Values

The values to be given on input statements for certain quantities can be calculated internally from other values and stored for later use. Values for the following quantities (and any equivalenced quantities) may be given in this manner:

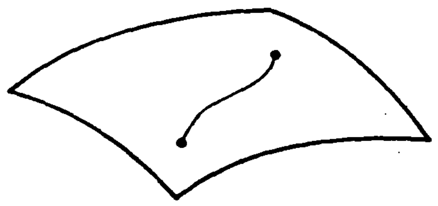
POINTS, POINTSI, POINT, START, END

A negative value given for any of these quantities causes a value to be obtained from the storage location indicated by its magnitude. The stored value is established by a prior use of ITEM = "SETVAL" (Section I-E23).

Internally calculated values (Section I-E23) can also be used for SPACE. Internally calculated integer values are set by ITEM="SETNUM", rather than "SETVAL", the latter being used for real values. A value can be simply stored by these operations also.

8. Curves on Curved Surfaces

Certain operations can be applied on curved surfaces, as well as in the plane.



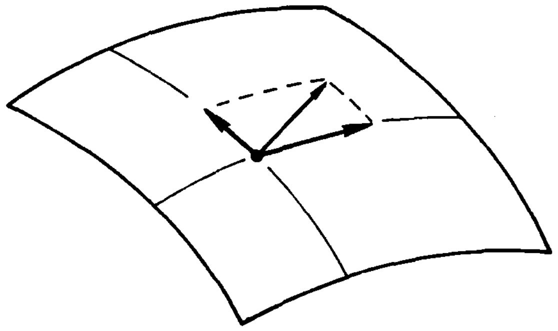
In this case the surface must have been generated by any of the surface-generation operations and splined by the use of the operation ITEM = "SPLINE" before the present operation is invoked. This mode is activated by including

SURFACE = "CURVED"

on the input statement for the present operation. The operations that can be used in this manner are the following:

LINE, SCURVE, TRANSUR, TENSUR, BLEND, PATCH

In this mode, values given on input for the Cartesian coordinates of an end point are converted by the code to those of the closest point on the surface. Similarly, slope vectors given on input are projected onto tangents to the surface:



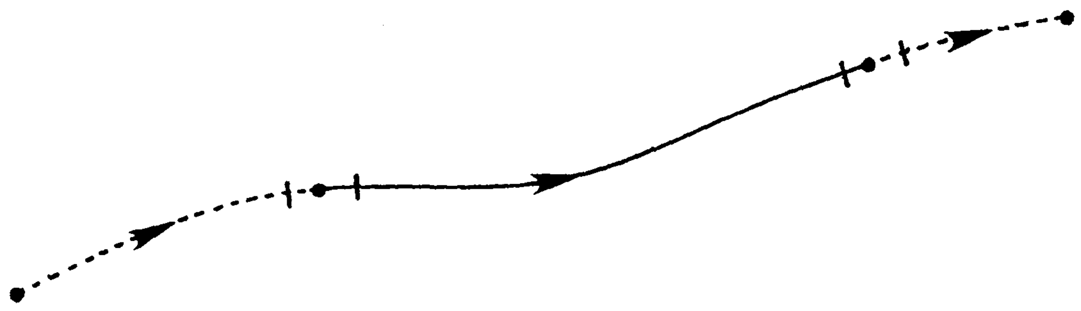
9. Numbered Points

Points can be addressed by numbers if the operation ITEM="POINT" (Section I-E25) has been used to set the Cartesian coordinates of the point. In that case a single positive integer given for R1 or R2 will be taken to be the point number, and the corresponding coordinates will be retrieved from storage.

10. Spacing from Existing Curve

The spacing given in SPACE (or its equivalenced quantities) can be taken from that on an end of an existing curve by giving a positive integer for SPACE with RELATIV omitted. The number given identifies a numbered segment (a segment number is the COREOUT number given when the curve was generated, Section I-B2) from which the spacing is taken. Spacings specified in this manner are by nature absolute.

If END(i)="FIRST" is included, the value for SPACE(i) is taken from the first end of the existing curve, otherwise if END(i)="LAST", SPACE(i) is from the last end. Here i=1 corresponds to the spacing at the first end of the curve being constructed, and 2 to the last end. The default for END is "LAST","FIRST". Although the existing curves do not have to be contiguous with the one being constructed, such is a common case and this default then corresponds to the most likely usage:



11. Scratch Files

The code uses files 7-10 as scratch files.

C. GENERATION OF CURVES

1. <u>Plane Conic-Section Curves</u>

The operation

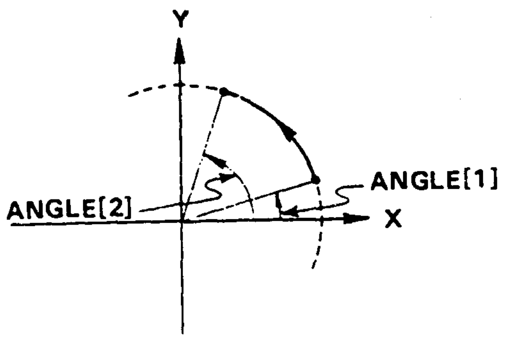
ITEM = "CONICUR"

generates a conic-section curve in the x-y plane (i.e., z=0). The curve may be a closed circle or ellipse, or may be an arc segment of a circle, ellipse, parabola, or hyperbola.

The number of points on the curve is given as

POINTS = number of points

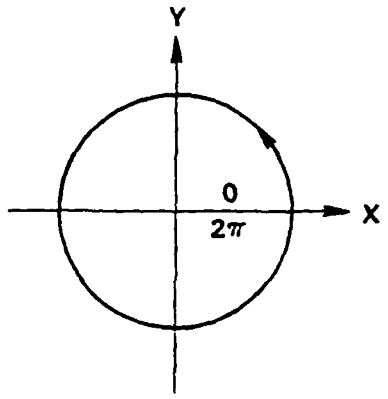
(The first and last points on closed curves, though coincident, are each counted.) The locations of the end points of the curve are set by the specification of two angles, with the points progressing from the first angle to the second:



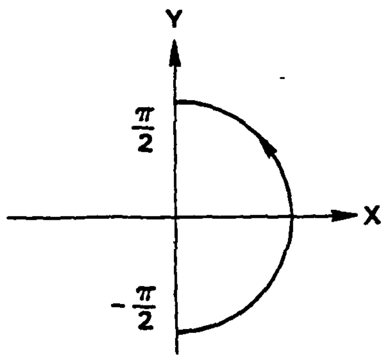
These angles are measured counterclockwise from the positive x-axis, and are given in degrees by

ANGLE = first angle, second angle

The first angle may exceed the second, in which case the point progression is clockwise, and negative angles are allowed. If ANGLE is not included, these angles will default to 0 and 2π for the closed curves:



and to -π/2 and π/2 for the arc segments:



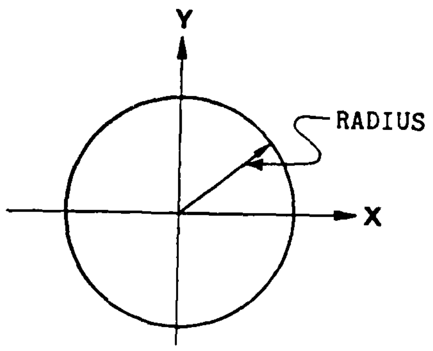
The angular distribution of the points can be set by including DISTANG and the associated distribution parameters (Section I-B6). Otherwise, the points will be placed at equal angular spacings. Spacings given here by SPACANG, and an interior arc length given by ARCIANG, must be fractions of the total angle swept by the curve.

The type of curve is set by

TYPE = "type of curve"

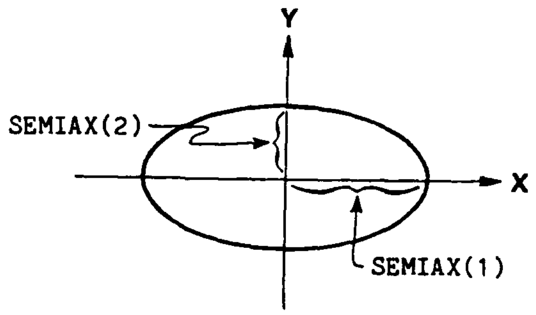
and the six possibilities are listed below, together with the relevant quantities to be included in each case:

TYPE = "CIRCLE" - closed circle centered at origin:



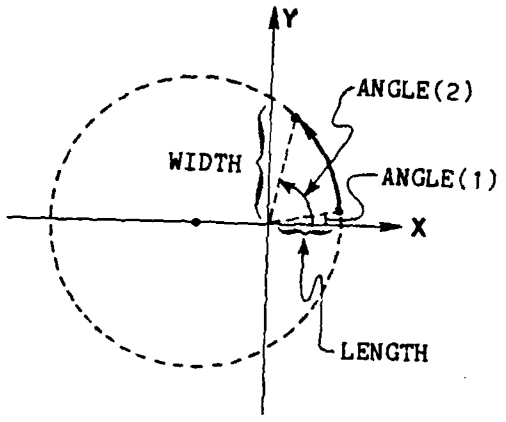
RADIUS = radius

TYPE = "ELLIPSE" - closed ellipse centered at origin:



SEMIAX = x semi-axis, y semi-axis

TYPE = "CIRARC" - arc of circle with center on x-axis:



LENGTH = positive x-intercept*

WIDTH = positive y-intercept*

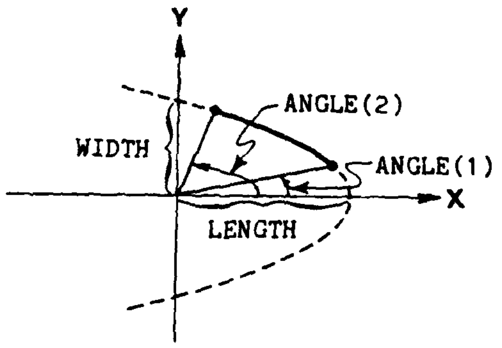
TYPE = "ELLIARC" - arc of ellipse with one axis on x-axis (see figure above).

LENGTH = positive x-intercept*

WIDTH = positive y-intercept*

ECCENT = eccentricity of ellipse

TYPE = "PARABOLA" - arc of parabola with axis on x-axis and vertex on positive x-axis:



LENGTH = positive x-intercept*

WIDTH = positive y-intercept*

----

*These intercepts refer to the complete curve from which the segment defined by the two angles is taken.

LENGTH = positive x-intercept*

WIDTH = positive y-intercept*

TYPE = "HYPERBOL" - arc of hyperbola with axis on x-axis and vertex on positive x-axis (see figure above).

LENGTH = positive x-intercept*

WIDTH = positive y-intercept*

SYMTOT = asymptote angle in degrees

Finally, the curve can be stored by the inclusion of COREOUT or FILEOUT (Section I-B2), and can be printed and/or plotted by including OUT, together with the associated output parameters (Section I-B5).

2. Cubic Space Curve Between Two Points

A cubic curve between two points, with specified slope vector at each end, can be generated by the operation

ITEM = "SCURVE"

The number of points on the curve is given as

POINTS = number of points

The end points are given by including

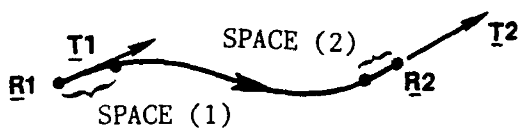
R1 = three Cartesian coordinates of first point

R2 = three Cartesian coordinates of last point

The unit tangents at the ends are given as

T1 = three direction cosines of unit tangent at first point

T2 = three direction cosines of unit tangent at last point



The point progression is from R1 to R2, and the tangents must be directed accordingly. The direction cosines of the unit tangents can be given either as actual cosines, or as the angles (in degrees) of which the cosine is to be taken. (Values greater than one in magnitude are taken as angles, while magnitudes of one or less are taken as cosines.) The spacings at each end of the curve are given as

SPACE = spacing at first end, spacing at second end

The point distribution on the curve is set automatically according to the hyperbolic tangent form, Section I-B6, so that no other distribution parameters are involved.

A single positive integer for R1 or R2 is taken as a point number, set by a previous usage of ITEM="POINT" (Section I-B9). It is not necessary to use the same mode for both R1 and R2. The spacing can be set in several ways as discussed in the revision of Section I-B6.

Either or both of the end points of the curve may be set by prior usages of the operation ITEM="GETEND" (Section I-E13), and both end points and unit tangents thereat may be set by prior usages of the tab operations "GENTAB" (Section I-E9) or "CURTAB" (Section I-E11). With such prior setting, the corresponding R1,R2,T1, or T2 is omitted from the present operation. Both spacings, however, are still required. (A cubic curve with the spacing also set from another curve or surface can be generated by the operation ITEM = "PATCH", Section I-D3)

A cubic curve on a curved surface can be generated by this operation with

SURFACE = "CURVED"

included (Section I-B8).

Finally, the curve can be stored by the inclusion of COREOUT or FILEOUT (Section I-B2), and can be printed and/or plotted by including OUT, together with the associated output parameters (Section I-B5).

3. Cubic Space Curve Connecting Two Surfaces(Curves)

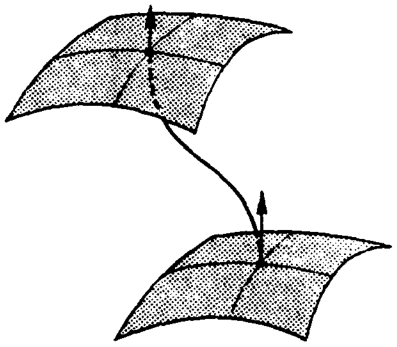
A cubic space curve connecting two existing surfaces(curves), with the slope at each end of the connecting curve matching either a normal or a tangent to the existing surfaces(curves), can be generated using the operation

ITEM = "PATCH"

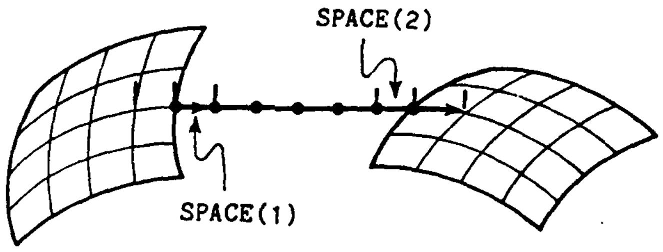
The number of points on the cubic curve is given by

POINTS = number of points

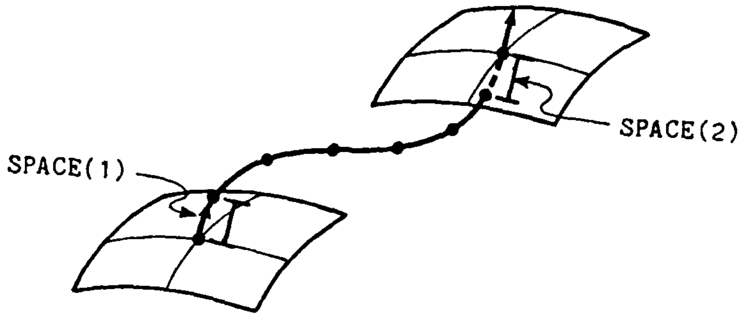
The end points, called tab points here because a slope vector (tab) is associated with each, must have been generated by two usages of the operations ITEM="GENTAB" (Section I-E9) or "CURTAB" (Section I-E11), or must have been input directly using the operation ITEM="SETTAB" (Section I-E12). The former select a point on an existing surface or curve, respectively, as the tab point, and set the lope vector there as either a normal or a tangent to that surface(curve). The latter operation sets the tab point and the slope vector directly by input. The two tab points can be set by any combination of these three operations. The points on the cubic curve run from the first tab point set to the other.



The spacing on each end of the cubic curve can be continuous with that on the surface from which the tab was taken if the tabs are tangents.



For tabs that are normals,



the spacing must be specified by including

SPACE = spacing at first end, spacing at second end

This spacing will be taken as relative unless

RELATIVE = "NO", "NO"

is included. With relative spacing at either end

TOTARC = total arc length

must also be included.

Note that if only the spacing at the second end is to be specified, the form is

SPACE(2) = spacing

and

RELATIV(2) = "NO"

while that at only the first end can be given simply as

SPACE = spacing

and

RELATIV = "NO"

The spacing can also be directly specified for tabs that are tangents by including SPACE, to override the continuous spacing.

This operation can be applied on a curved surface (Section I-B8) by including

SURFACE = "CURVED"

Finally, the curve can be stored by the inclusion of COREOUT or FILEOUT (Section I-B2), and can be printed and/or plotted by including OUT, together with the associated output parameters (Section I-B5).

4. Straight Line

A straight line between two points can be generated by the operation

ITEM = "LINE"

The number of points on the line is given by

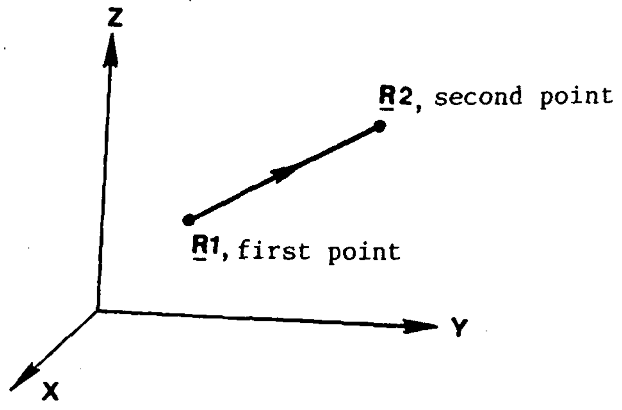
POINTS = number of points

The end points are given by including

R1 = three Cartesian coordinates of first point

R2 = three Cartesian coordinates of last point

Alternatively, either, or both, of these end points can have been set by prior use of the operation ITEM="GETEND" (Section I-E9) which selects a point from an existing curve.



The point progression on the line is from R1 to R2.

The distribution of the points on the line can be set by including DISTYP and the associated distribution parameters (Section I-B6). Otherwise the points will be equally spaced. The spacing given here by SPACE, and an arc length given by ARCINT, may be fractions of the total line length, or may be actual distances (with RELATIV="NO" included).

A single positive integer for R1 or R2 is taken as a point number, set by a previous usage of ITEM="POINT" (Section I-B9). It is not necessary to use the same mode for both R1 and R2. The spacing can be set in several ways as discussed in the revision of Section I-B6.

This operation can be applied on a curved surface (Section I-B8) by including

SURFACE = "CURVED"

Finally, the line can be stored by the inclusion of COREOUT or FILEOUT (Section I-B2) and can be printed and/or plotted by including OUT, together with the associated output parameters (Section I-B5).

5. Generation of a Curve as the Intersection of Two Surfaces

The operation

ITEM = "INTSEC"

generates a curve formed by the intersections of one family of curves on an intersecting surface with an intersected surface. Both of these surfaces must have been generated prior to the invocation of the present operation. The intersecting surface must be in current position, (Section I-E16), and the storage location of the intersected surface is given by COREIN or FILEIN (Section I-B3). The intersecting curves are those in the second direction on the intersecting surface, (all of which must intersect the intersected surface), so that the number of points on the intersection curve will be the same as the first dimension of the intersecting surface. Finally, the line can be stored by the inclusion of COREOUT or FILEOUT (Section I-B2), and can be printed and/or plotted by including OUT, together with the associated output parameters (Section I-B5).

D. GENERATION OF SURFACES

1. <u>Flat Conic-Section Surfaces</u>

A flat surface on the x-y plane (with z=0) bounded by a conic-section curve (and straight lines closing open curves) can be generated by the operation

ITEM = "FLATSUR"

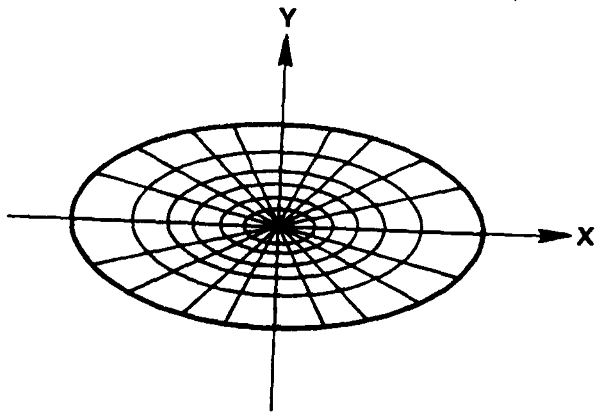
The conic-section curve may be any of the plane conic-section curves generated by the operation ITEM = "CONICUR" (Section I-C1). The points will be placed on radial and circumferential lines. The number of points on each of the two sets of lines is given by

CURPTS = number of circumferential points

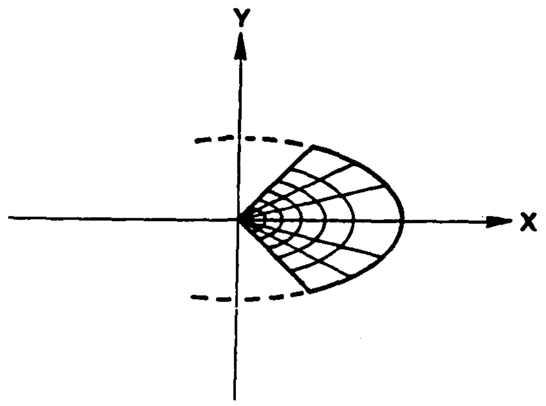
RADPTS = number of radial points

(The first and last points on closed lines, though coincident, are each counted.)

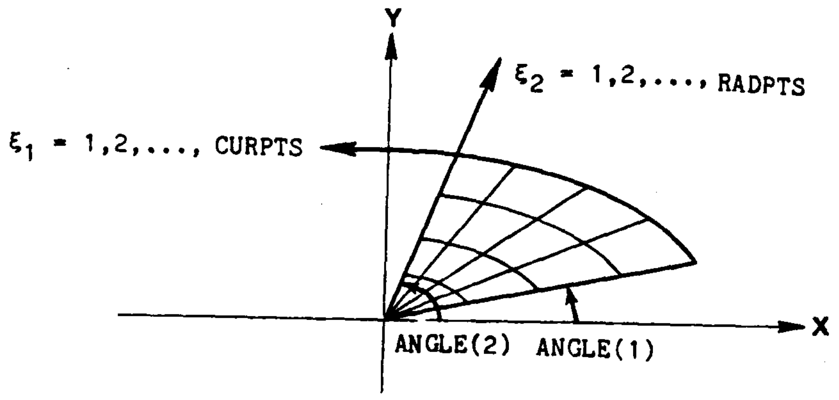
The parameters, including ANGLE, for the conic-section curve are the same as described for ITEM="CONICUR". The closed curves give flat surfaces of the form



while the open segments are closed by straight lines from the ends of the curves to the origin:



On the resulting surface, the first subscript runs along the circumferential curves, so that the surface dimensions are (CURPTS,RADPTS). The points on the circumferential curves run from the first angle to the second, as with ITEM="CONICUR", while the points on the radial lines run from the origin to the conic-section curve:



The angular distribution of points on the circumferential lines can be set as with ITEM="CONICUR", using DISTANG, etc. (Section I-B6); otherwise the angular distribution will be uniform. The spacings, and an interior arc length, here are relative to the total angle. The radial point distribution can be set by including DISTRAD, etc. (Section I-B6); otherwise, the radial points will be equally spaced. The spacings, and an interior arc length, for the radial distribution are fractions of the local radial distance to the curve.

Finally, the surface can be stored by the inclusion of COREOUT or FILEOUT (Section I-B2), and can be printed and/or plotted by including OUT, together with the associated output parameters (Section I-B5).

2. Conic-Section Surfaces

The operation

ITEM = "CONISUR"

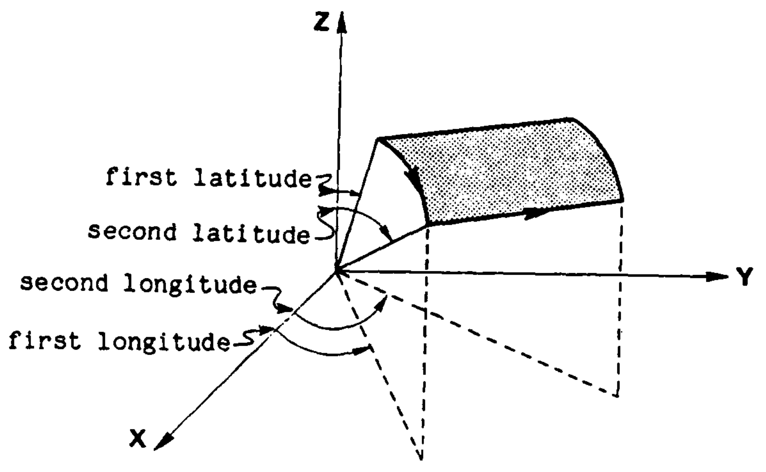
generates a conic-section surface, which may be a closed sphere or ellipsoid, or may be a segment of a sphere, ellipsoid, elliptical cone, or elliptical paraboloid. The points will be placed on the surface along the lines swept out by the two polar coordinate angles (latitude and longitude lines for the sphere, and analogous lines in the other cases.) The number of points on each of the two sets of lines is given by

LONPTS = number of longitude points

LATPTS = number of latitude points

(The first and last points on closed lines, though coincident, are each counted.)

The locations of the end points of the two sets of lines are determined by the specification of two angles in each case, with the points progressing from the first angle to the second:



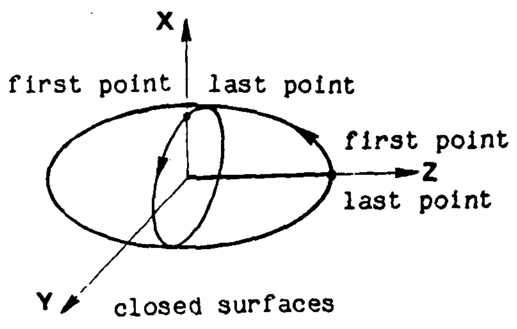
The latitude angle is measured from the positive z-axis, and ranges from 0 to π in magnitude. The longitude angle is measured from the positive x-axis toward the positive y-axis, and ranges from 0 to 2π in magnitude. The angles are given in degrees by

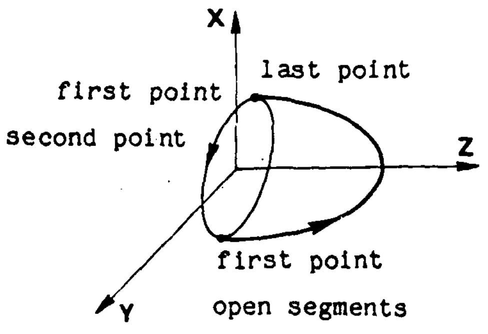
LON = first longitude angle, second longitude angle

LAT = first latitude angle, second latitude angle

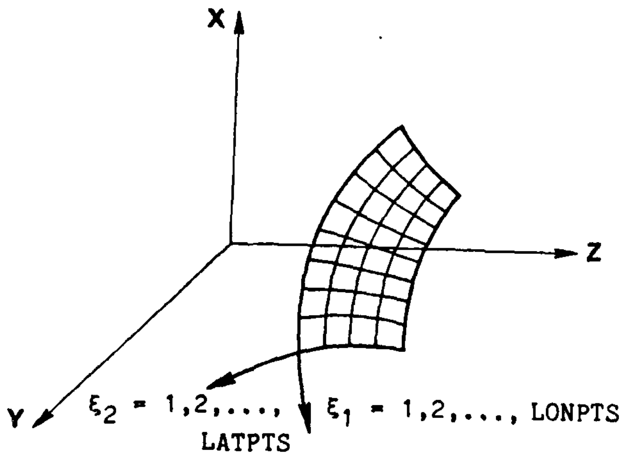
The first angle of either pair may exceed the second, the progression of points being from the first angle to the second in any case, and negative longitudes are allowed.

If LON is not included, then the longitude limits will default to 0 and 2π, and if LAT is omitted the latitude limits will default to 0 to π for the closed surfaces and to 0 to $\pi/2$ for the open segments:





The first subscript on the surface runs along the lines of constant latitude, so that the surface dimensions are (LONPTS,LATPTS):



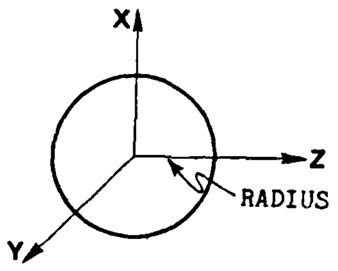
The angular distributions of the points on each set of lines can be set (Section I-B6) by including DISTLON, etc., for longitude, and DISTLAT, etc., for latitude. Otherwise, the points will be placed at equal angular increments. Spacings and interior arc lengths given here must be fractions of the total angle swept by the points on the corresponding set of lines.

The type of surface is set by

TYPE = "type of surface"

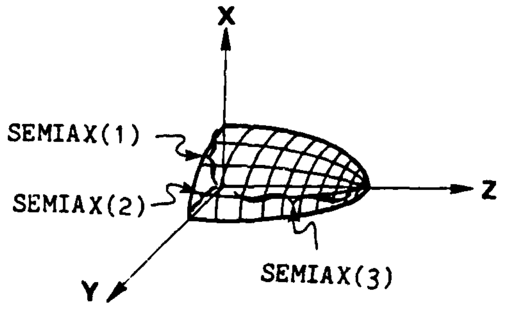
and the six possibilities are listed below, together with the relevant parameters to be included in each case:

TYPE = "SPHERE" - closed sphere centered at origin:



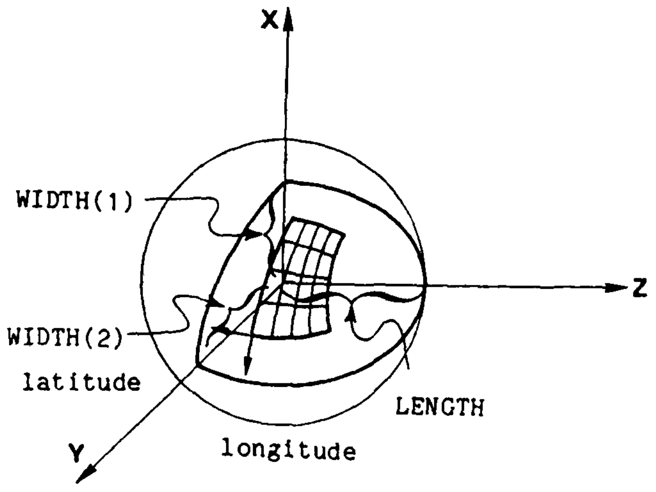
RADIUS = radius

TYPE = "ELIPSOID" - closed ellipsoid centered at origin:



SEMIAX = x semi-axis, y semi-axis, z semi-axis

TYPE = "SPHSEG" - segment of sphere with center on z-axis:



LENGTH - positive z-intercept*

LENGTH = positive z-intercept*

WIDTH = positive x-intercept,* positive y-intercept*

TYPE = "ELLISEG" - segment of ellipse with one axis on z-axis (see figure above).

LENGTH = positive z-intercept*

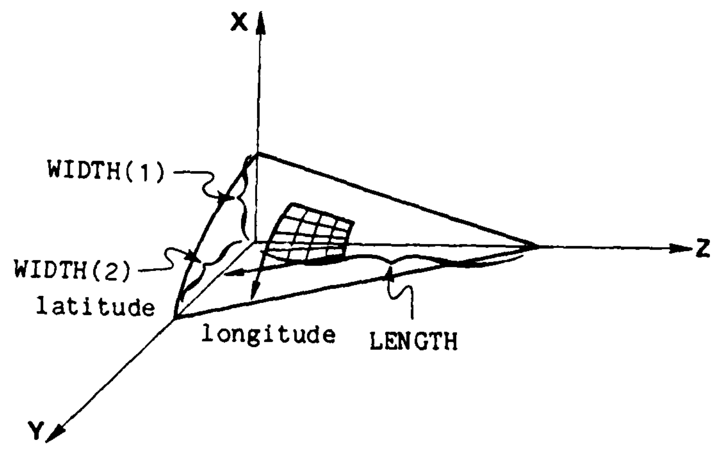
WIDTH = positive x-intercept*, positve y-intercept*

ECCENT = eccentricity of ellipsoid

TYPE = "ELLICONE" - segment of elliptical cone with axis on z-axis and vertex on positive z-axis:

----

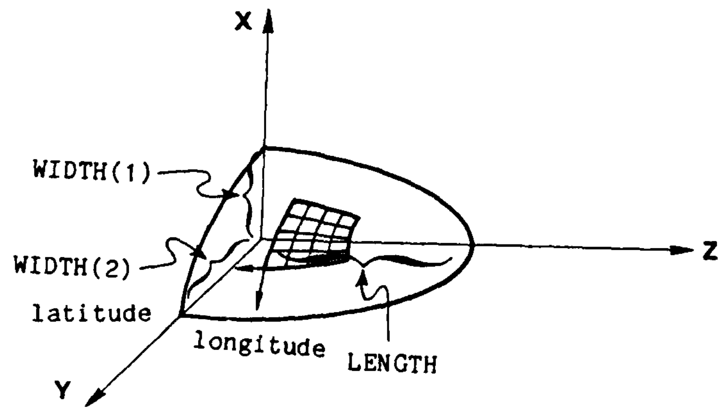
*The intercepts are those of the complete surface from which the segment is taken.



LENGTH = positive z-intercept*

WIDTH = positve x-intercept*, positve y-intercept*

TYPE = "ELLIPAR" - segment of elliptic paraboloid with axis on z-axis and vertex on positve z-axis:



LENGTH = positive z-intercept*

WIDTH = positve x-intercept*, positve y-intercept*

Finally, the surface can be stored by the inclusion of COREOUT or FILEOUT (Section I-B2), and can be printed and/or plotted by including OUT, together with the associated output parameters (Section I-B5).

3. Cubic Patch Surface

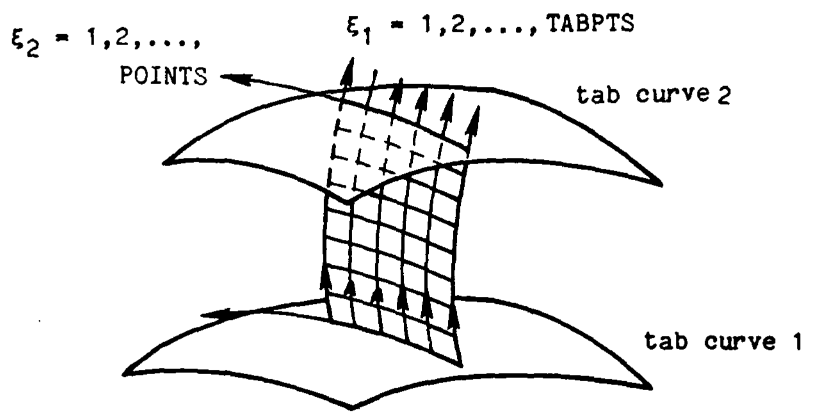
A surface composed of cubic space curves connecting corresponding points on two end curves, with slope vectors specified on the end curves, can be generated with the operation

ITEM = "PATCH"

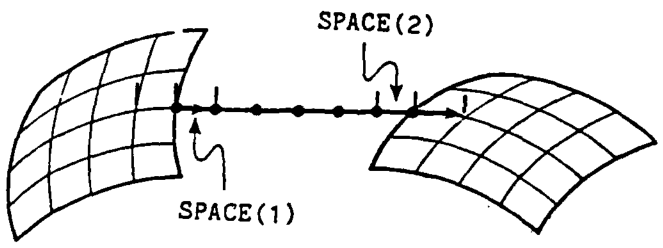
The number of points on the cubic curves, and that on the end curves, is given by

POINTS = number of points on cubic curves

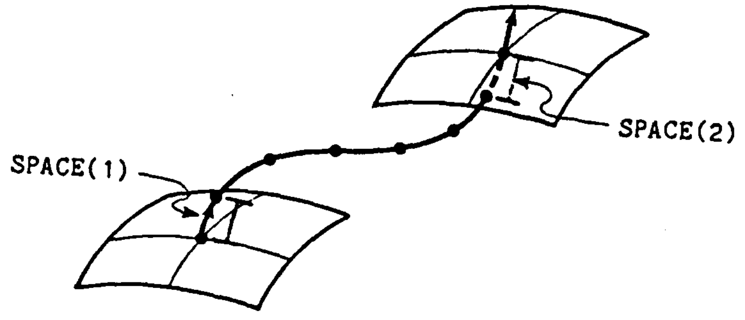
The end curves, called tab curves here because a slope vector (tab) is associated with each point thereon, must have been generated by two usages of the operations ITEM="GENTAB" (Section I-E9), "EDGETAB" (Section I-E10), "CURTAB" (Section I-E11), or "SETTAB" (Section I-E12). These operations select a portion of a grid line on an existing surface as the tab curve, and set the slope vectors thereon as either a normal or one of the tangents to that surface. The operation ITEM="PATCH" thus can serve to generate a surface connecting two existing surfaces. The cubic surface generated can, of course, be treated as a separate surface also, with the two existing surfaces having served only as sources from which to extract the tab curves. On the cubic surface, the first subscript is along the tab curves, with the second along the cubic connecting curves, so that the dimensions of the surface are (TABPTS,POINTS), where TABPTS, the number of points on the tab curves, is obtained from the tab curves. The points on the connecting curves run from the first tab curve designated to the other:



The spacing on each end of the cubic curve can be continuous with that on the surface from which the tab curves were obtained if the tabs are tangents.



For tabs that are normals,



the spacing must be specified by including

SPACE = spacing at first end, spacing at second end

This spacing will be taken as relative unless

RELATIVE = "NO", "NO"

is included. With relative spacing at either end

TOTARC = total arc length

must also be included.

Note that if only the spacing at the second end is to be specified, the form is

SPACE(2) = spacing

and

RELATIV(2) = "NO"

while that at only the first end can be given simply as

SPACE = spacing

and

RELATIV = "NO"

The spacing can also be directly specified for tabs that are tangents by including SPACE, to override the continuous spacing.

This operation can be applied on a curved surface (Section I-B8) by including

SURFACE = "CURVED"

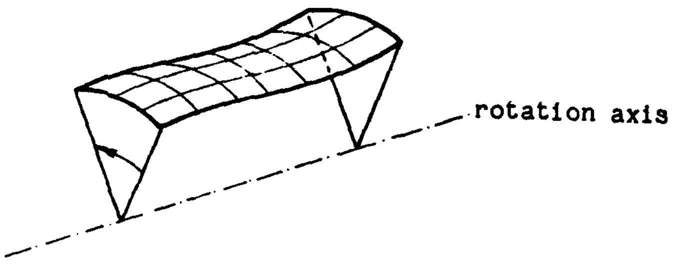
Finally, the surface can be stored by the inclusion of COREOUT or FILEOUT (Section I-B2), and can be printed and/or plotted by including OUT, together with the associated output parameters (Section I-B5).

4. Generation of a Surface by Rotation of a Curve

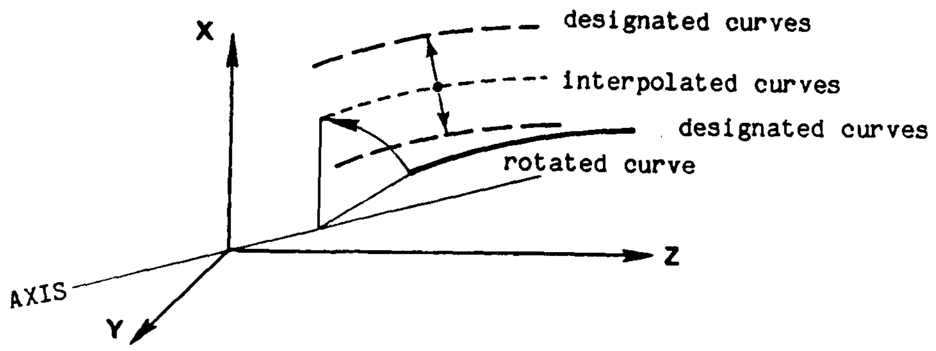
The operation

ITEM = "ROTATE"

generates a surface by rotating a space scurve about an axis:



The surface is formed by a set of curves, each of which results from rotation of a given curve to successive angular positions. The given curve that is rotated may be different for each rotation angle and is determined by interpolation between two existing curves before rotation. These two existing curves must have been generated in the same axes system, with the same number of points, and are designated by one or two usages of the operation ITEM="BOUNCUR" (Section I-E5) before the present operation is invoked. The curves that are rotated to the first and last angular positions, respectively, coincide with the first and second designated curves. Note that the interpolation is done before the rotation.

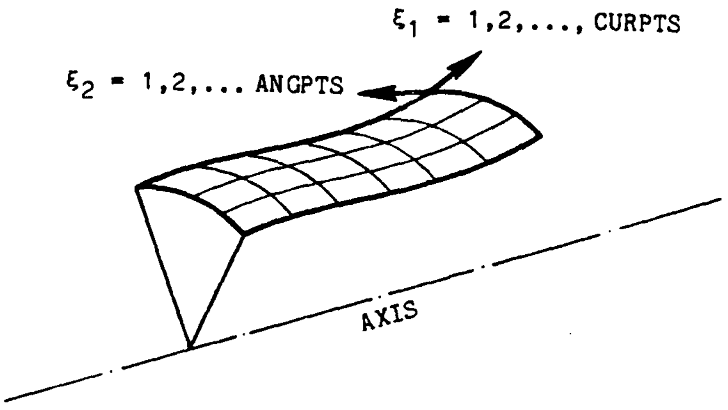


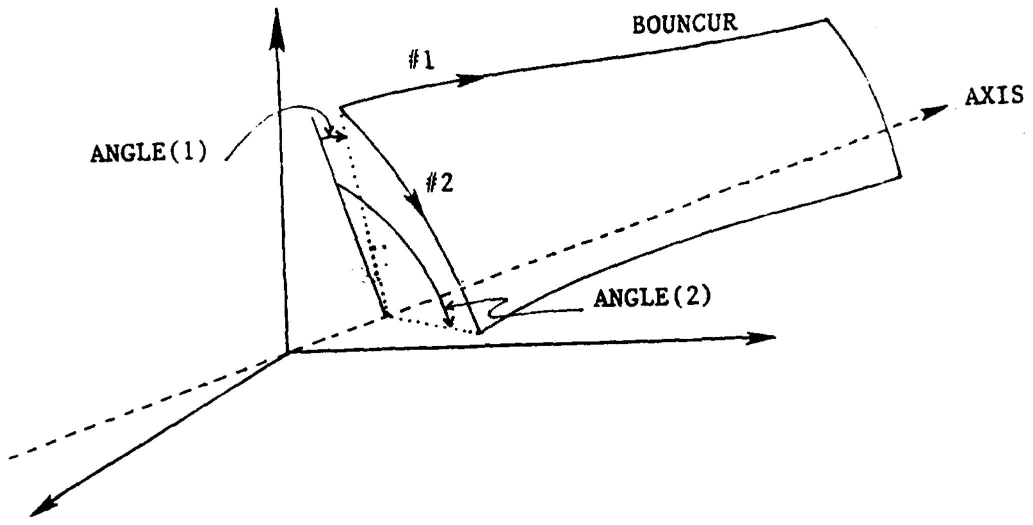
These two designated curves must both have this number of points. If only one use of "BOUNCUR" is made, the same curve will be rotated to each angular position.

The number of rotation angle positions is given by

ANGPTS = number of rotation angles

On the surface, the first subscript runs along the rotated curves, and the second runs in the rotation direction. The dimensions of the surface are thus (CURPTS,ANGPTS), where CURPTS is the number of points on the rotated curves.

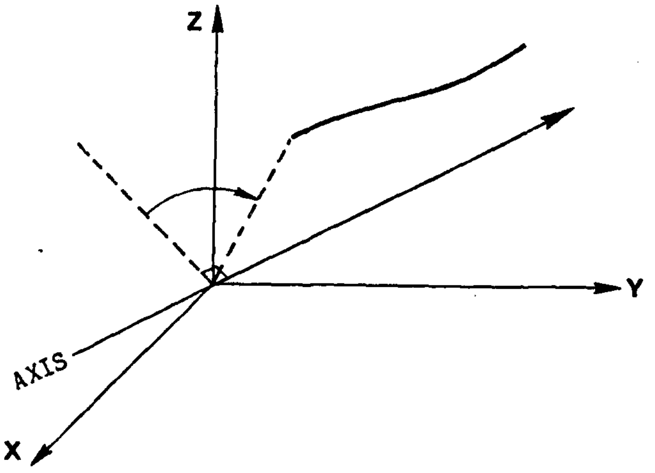




The rotation axis is set by

AXCOS = three direction cosines of axis

and passes through the origin of the axes system in which the two designated curves are defined. The entries given may be actual cosines or may be the corresponding angles (in degrees). The direction cosines are relative to the axes system in which the two designated curves are defined. The rotation angle is positive clockwise looking down the axis:



The axis is defaulted to the z axis. The limits of the rotation are set by

ANGLE = initial rotation angle in degrees, final angle

If only one angle is given, the rotation will be a full 360° starting from that angle. (The first and last angles in this case are still counted separately even though they are coincident.) Either limit may be the larger, the rotation being from the first angle to the second in any case, and negative angles are allowed.

The distribution of the rotation angles can be set by including DISTANG, etc. (Section I-B6). Otherwise, the angles will be equally spaced. Spacings and arc length are given here as fractions of the difference between the rotation angle limits.

A distribution can also be set for the interpolation between the two designated curves by including DISTCUR, etc. If no distribution is set, the interpolation is linear by the number of the rotation position. Spacings and arc lengths here are fractions of the total number of rotation positions. Another type of distribution for the interpolation can also be used as follows: If DISTCUR="CURVE", the fractions for the interpolation will be taken from a distribution function to be supplied. This function is of the form y(x), where the abscissa is the angle and the ordinate is a characteristic length scale. (Both of these will be properly normalized by the code and thus do not have to conform to specific ranges as input.) This distribution function can be generated by any of the curve generation operations, or read in, but it must be defined in the x-y plane and must be single-valued in x. It does not have to be monotonic. This distribution function could, for example, be a radius that is a function of the rotation angle. The number of points for this distribution function is given as

POINTSI = number of points

and its location must be given by COREIN or FILEIN (Section I-B3).

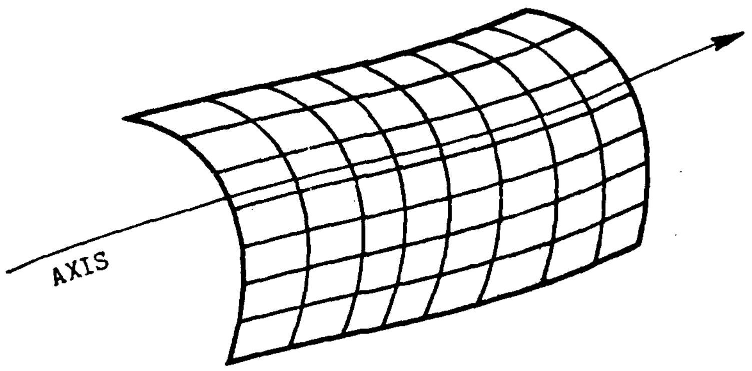
Finally, the surface can be stored by the inclusion of COREOUT or FILEOUT (Section I-B2), and can be printed and/or plotted by including OUT, together with the associated output parameters (Section I-B5).

5. Generation of a Surface by Stacking Curve

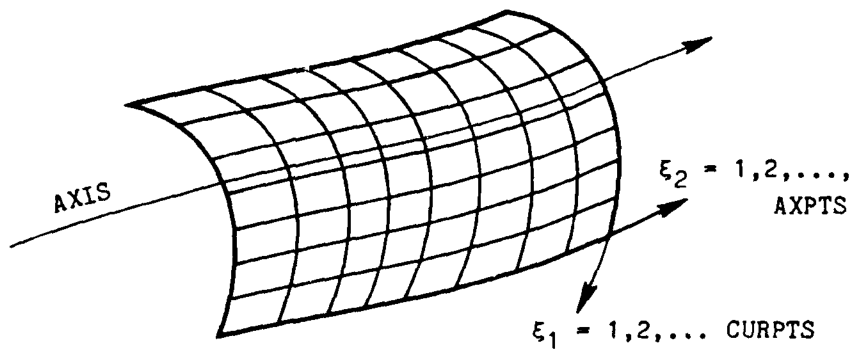
The operation

ITEM = "STACK"

generates a surface by stacking space curves at successive positions along an (possibly curved) axis:



At each point on the axis curve, the curve that is placed there may be different for each position, and is determined by interpolation between two existing curves before being placed. These two existing curves must have been generated in the same axes system, with the same number of points, and are designated by one or two usages of the operation ITEM="BOUNCUR" (Section I-E5) before the present operation is invoked. If only one use of "BOUNCUR" is made, the same curve will be placed at each point on the axis. The axis curve must also be set prior to the present operation by use of ITEM="AXIS" (Section I-E6). The curves that are placed at the first and last positions, respectively, on the axis coincide with the first and second designated curve. Note that the interpolation is done before the placement. On the surface, the first subscript runs along the stacked curves, and the second runs down the axis. The dimensions of the surface are thus (CURPTS,AXPTS):



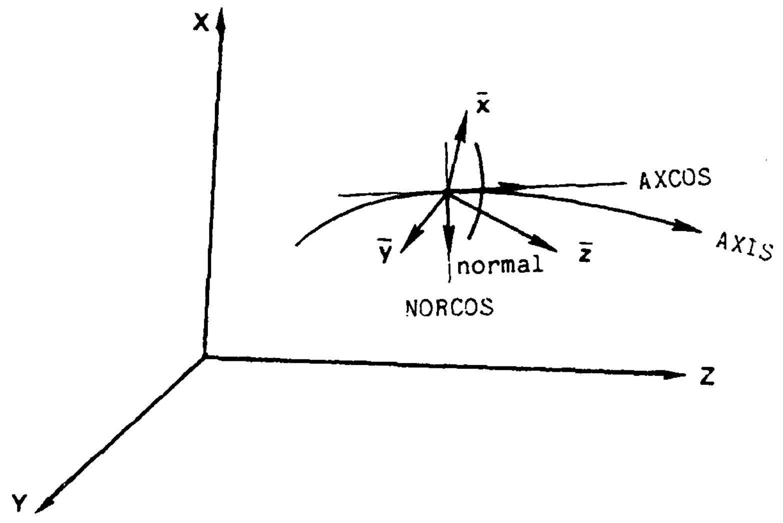
The direction of the axis of the stack in the axes system in which the existing curves are defined is set by

AXCOS = three direction cosines of axis

The direction of the principal normal to the axis in this axes system is defined by

NORCOS = three direction cosines of normal

(For each of these, the entries given may be actual cosines or may be the angles in degrees.) The axes system containing the curve is placed with its origin at each successive axial position, and with the specified axis direction and normal coincident with the local tangent and principal normal to the axis curve:



The axis and normal are defaulted to the z and x axes respectively.

The distribution can be set by the interpolation between the two existing curves for the curve to be placed by including DISTCUR, etc, (Section I-B6). If no distribution is set, the interpolation is linear by point number on the axis. Spacings and arc length here are fractions of the total number of points on the axis. Two other types of distribution for the interpolation can also be used. If DISTCUR="CURVE", the fractions for the interpolation will be taken from a distribution function to be supplied. This function is of the form y(x), where the abscissa is the angle and the ordinate is a characteristic length scale. (Both of these will be properly normalized by the code and thus do not have to conform to specific ranges as input.) This distribution function can be generated by any of the curve generation operations or read in, but it must be defined in the x-y plane and must be single-valued in x. It does not have to be monotonic. This distribution function could, for example, be a radius that is a function of the rotation angle. The number of points on this distribution curve is given as

POINTSI = number of points

and its location must be given by COREIN or FILEIN (Section I-B3). If DISTCUR="ARC" the interpolation will be linear with arc length on the axis.

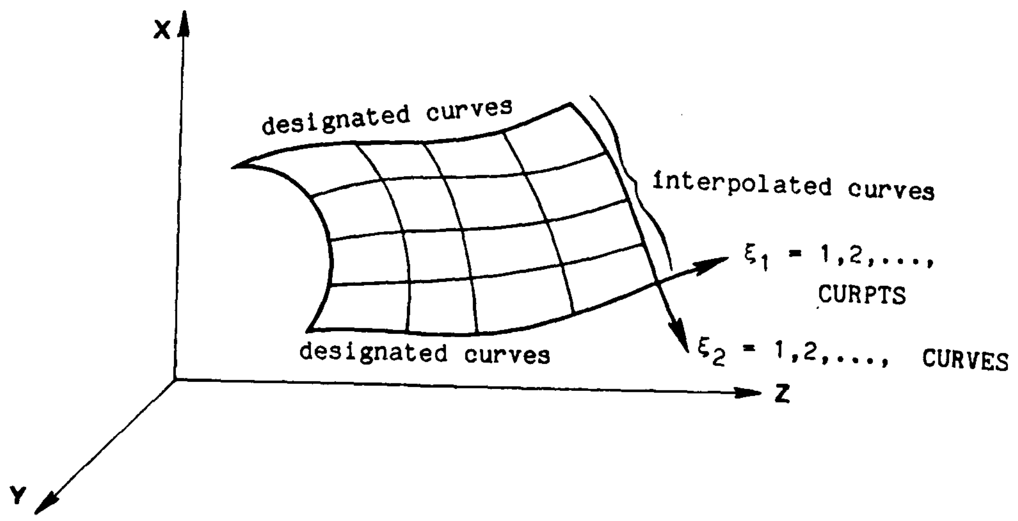
Finally, the surface can be stored by the inclusion of COREOUT or FILEOUT (Section I-B2), and can be printed and/or plotted by including OUT, together with the associated output parameters (Section I-B5).

6. Generation of a Surface by Blending Curves

The operation

ITEM = "BLEND"

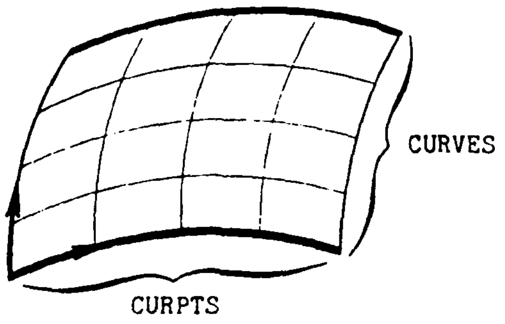
generates a surface by interpolation between two designated existing space curves:



These two existing curves must have been generated in the same axes system and must have the same number of points. These two curves are designated by one or two usages of the operation ITEM="BOUNCUR" (Section I-E5) before the present operation is invoked. The number of curves to be interpolated (including the two end curves) is given by

CURVES = number of interpolated curves

On the surface, the first subscript runs along the interpolated curves, and the second runs from the first interpolated curve to the last. The dimensions of the surface are thus (CURPTS,CURVES):



The distribution can be set for the interpolation by including DISTCUR, etc., (Section I-B6). If no distribution is set, the interpolation is linear by curve number. Spacings and arc length here are fractions of the total number of curves. Another type of distribution for the interpolation can also be used as follows: If DISTUR="CURVE", the fractions for the interpolation will be taken from a distribution function to be supplied. This function is of the form y(x), where the abscissa is the angle and the ordinate is a characteristic length scale. (Both of these will be properly normalized by the code and thus do not have to conform to specific ranges as input.) This distribution function can be generated by any of the curve generation operations or read in, but it must be defined in the x-y plane and must be single-valued in x. It does not have to be monotonic. This distribution function could, for example, be a radius that is a function of the rotation angle. The number of points on this distribution curve is given as

POINTSI = number of points

and its location must be given by COREIN or FILEIN (Section I-B3).

This operation can be applied on a curved surface (Section I-B8) by including

SURFACE = "CURVED"

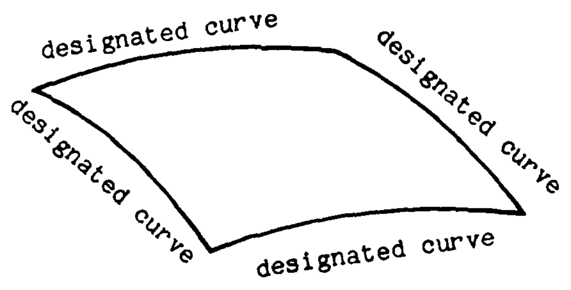
Finally, the surface can be stored by the inclusion of COREOUT or FILEOUT (Section I-B2), and can be printed and/or plotted by including OUT, together with the associated parameters (Section I-B5).

7. Generation of a Surface by Transfinite Interpolation

The operation

ITEM = "TRANSUR"

generates a surface by transfinite interpolation from four designated existing space curves:



These four edge curves must have been generated in the same axis system; opposite pairs must have the same number of points; and the ends must meet. These four curves are designated by four usages of the operation ITEM="EDGECUR" (Section I-E7) before the present operation is invoked.

This operation can be applied on a curved surface (Section I-B8) by including

SURFACE = "CURVED"

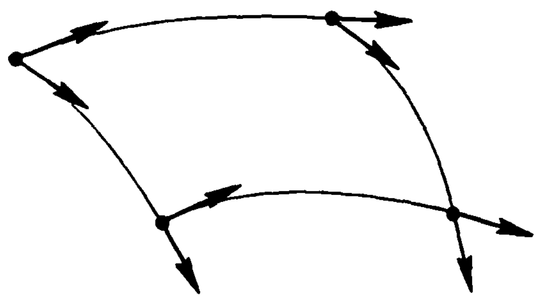
Finally, the surface can be stored by the inclusion of COREOUT or FILEOUT (Section I-B2), and can be printed and/or plotted by including OUT, together with the associated parameters (Section I-B5).

8. Generation of a Tensor-Product Surface (Coon's Patch)

The operation

ITEM = "TENSUR"

generates a surface by bi-cubic interpolation from four corners with specified slope vectors:



These four corners are set by four usages of the operation ITEM="SETCOR" (Section I-E8) before the present operation is invoked.

This operation can be applied on a curved surface (Section I-B8) by including

SURFACE = "CURVED"

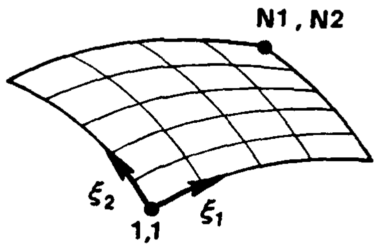
Finally, the surface can be stored by the inclusion or COREOUT or FILEOUT (Section I-B2), and can be printed and/or plotted by including OUT, together with the associated parameters (Section I-B5).

9. Generation of a Surface from a Surface-Parametric Coordinate Lattice

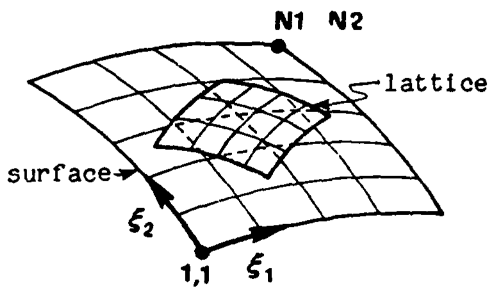
The operation

ITEM = "PARCOR"

generates a surface from an existing surface-parametric coordinate lattice placed on an existing surface. The lattice can have been generated by any of the 2D plane operations before the present operation is invoked. In the generation of the lattice the Cartesian coordinates used in the 2D plane operation are interpreted as the surface parametric coordinates. The SPLINE operation splines the existing surface in terms of the curvilinear coordinates, ξ_1 and ξ_2, that assume the integer values from 1 to the number of points in each direction at the points set on the surface. These curvilinear coordinates are the surface parametric coordinates, so that the range of values on the lattice must be within the range of these coordinates. For example, let the existing surface have N1xN2 points:



Then the lattice must be generated with "Cartesian" coordinates $x=\xi_1$ and $y=\xi_2$ within the ranges 1-N1 and 1-N2 by any of the 2D plane operations:



The surface on which the parametric coordinates are defined must also have been generated by any of the surface generation operations and splined by operation ITEM="SPLINE" before the present operation is invoked. Finally, the surface can be stored by the inclusion of COREOUT or FILEOUT (Section I-B2), and can be printed and/or plotted by including OUT, together with the associated parameters (Section I-B5).

E. AUXILLIARY OPERATIONS

1. Distributing Points on a Curve with Specified Spacing

Distribution of points on an existing curve with the relative or actual arc-length spacing specified at one or both ends, or at a specified point between the ends, is done by the operation

ITEM = "CURDIST".

The number of points to be placed on the curve is set by

POINTS = number

and the distribution is set by DISTYP and the associated distribution parameters (Section I-B6). The spacings given will be taken as actual arc lengths if

RELATIV = "NO"

is included. There are two entries in RELATIV, corresponding to the two in SPACE, when the spacing is specified on both ends. Similarly the second entry corresponds to the arc length for specified interior spacing. It is not necessary to include the total arc length with relative spacings here since it is known from the input curve. The spacing can be set in several ways as discussed in Section I-B6.

The points are distributed on a chord-length spline curve through the input point distribution. The default is a quadratic spline, i.e., with extrapolated curvature at the ends, but a natural spline (zero curvature ends) can be selected by including TYPE="NATURAL".

The curve on which the points are to be placed must have been generated prior to this operation. If this curve was generated by the preceding operation, it will automatically be available here. Otherwise, its storage location must be given by including COREIN or FILEIN (Section I-B3). Finally, if the point distribution is to be stored, the storage location must be given (Section I-B2), and printing and/or plotting can be called for (Section I-B5).

2. Distributing Points on a Curve According to Curvature

Distribution of points on an existing curve so that points are more closely spaced in regions of large curvature is done by the operation

ITEM = "CURVAT"

The number of points to be placed on the curve is set by

POINTS = number

The curve on which the points are to be placed must have been generated prior to this operation. If this curve was generated by the preceding operation, it will automatically be available here. Otherwise, its storage location must be given by including COREIN or FILEIN (Section I-B3). Finally, if the point distribution is to be stored, the storage location must be given (Section I-B2), and printing and/or plotting can be called for (Section I-B5).

3. Scaling a Surface(Curve)

A surface(curve) can be scaled by the operation

ITEM = "SCALE"

which multiplies the three Cartesian coordinates of each point by scale factors, there being a separate factor for each of the Cartesian directions. If the surface was not generated or treated by the preceding operation, its storage location must be given by including COREIN or

FILEIN (Section I-B3). This operation allows multiple surfaces to be subjected to simultaneous scaling by giving multiple entries for COREIN and/or FILEIN.

The three scale factors are given by

SCALE = x-factor, y-factor, z-factor

Negative values are permitted, causing mirror-image reversals. (Note that all three factors must be negative to preserve a right-handed system.) The scale factors can be given as quotients by including

SDENOM = x-denominator, y-denominator, z-denominator

SNUMER = x-numerator, y-numerator, z-numerator

instead of SCALE. Some factors can be input in one form and others in the other form. For example,

SCALE(1) = __, SCALE(3) = __, SDENOM(2) = __, SNUMER(2) = __

The scaled surface can be stored by including COREOUT or FILEOUT (Section I-B2), again perhaps with multiple entries, and can be printed and/or plotted by including OUT and the associated output parameters (Section I-B5).

4. Transforming a Surface(Curve)

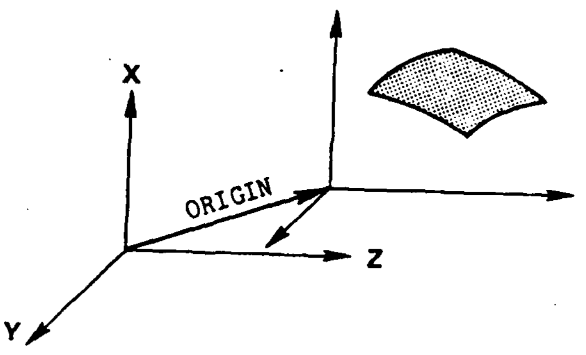
A surface(curve) can be translated, rotated, and/or scaled by the operation

ITEM = "TRANS"

If the surface was not generated or treated by the preceding command, its storage location must be given by COREIN or FILEIN (Section I-B3). This operation allows multiple surfaces to be subjected to the same transformation by giving multiple entries for COREIN and/or FILEIN.

Translation is done by the three components of the translation vector given in

ORIGIN = x-component, y-component, z-component

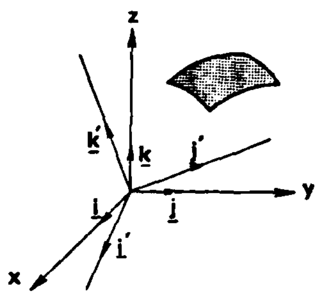


Rotation is done by giving either the 3x3 matrix (input by columns) of direction cosines as

COSINES =

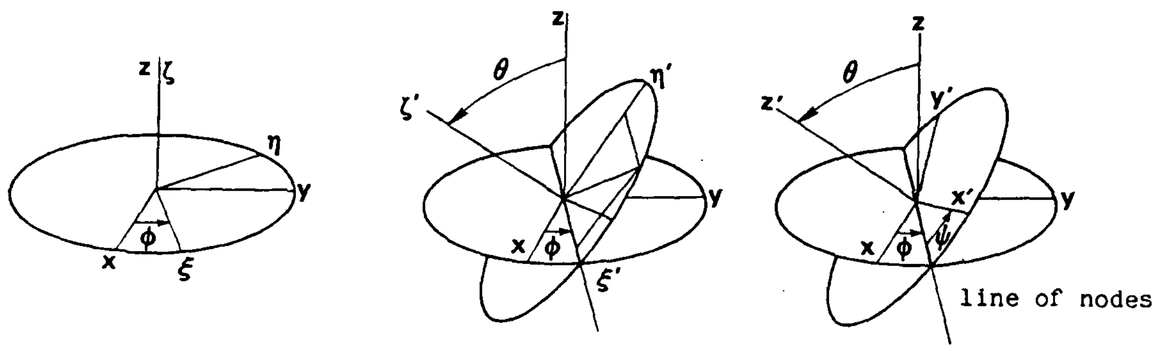
$$\begin{array}{lll} \underset{\sim}{i}\cdot\underset{\sim}{i}', & \underset{\sim}{i}\cdot\underset{\sim}{j}', & \underset{\sim}{i}\cdot\underset{\sim}{k}', \\ \underset{\sim}{j}\cdot\underset{\sim}{i}', & \underset{\sim}{j}\cdot\underset{\sim}{j}', & \underset{\sim}{j}\cdot\underset{\sim}{k}', \\ \underset{\sim}{k}\cdot\underset{\sim}{i}', & \underset{\sim}{k}\cdot\underset{\sim}{j}', & \underset{\sim}{k}\cdot\underset{\sim}{k}' \end{array}$$

where (i',j',k') is the axes system in which the curve is given.



Either actual cosines or the angles (in degrees) can be given. Alternatively, the rotation can be specified by giving the three Euler angles (in degrees) as

EULER = first rotation, second rotation, third rotation



Scaling is done by including SCALE, as described separately for ITEM="SCALE" (Section I-E3).

It is also possible to include an ad hoc coordinate change that has been inserted in the code by the user at the place marked

!!! AD HOC COORDINATE CHANGE !!!

in the main program. The ad hoc change is then activated by including

CHANGE = "YES"

In the transformation, the scaling is done first, followed by rotation, then translation, and finally the ad hoc change. The transformed surface can be stored by including COREOUT or FILEOUT (Section I-B2), again perhaps with multiple entries, and can be printed and/or plotted by including OUT and the associated output parameters (Section I-B5).

5. Designating a Curve to be Rotated, Stacked, or Blended

The generation of a surface by rotation, stacking, or blending of space curves by the operations ITEM="ROTATE" (Section I-D4), "STACK" (Section I-D5), or "BLEND" (Section I-D6) uses a curve that is interpolated between two existing curves which must be designated prior to the surface generation operation. Two existing curves are designated for this purpose by two usages of the operation

ITEM = "BOUNCUR"

The storage location of the existing curve must be given by COREIN or FILEIN (Section I-B3) unless it was generated by the operation immediately preceding the "BOUNCUR" operation.

In general, this operation will be used twice before the surface generation operation is used. The existing curve designated by the first use becomes the first of the two curves used in the surface generation, etc. If only one "BOUNCUR" operation is used, the single curve so designated is used for both curves, in the surface generation operation i.e., the interpolation is irrelevant, and thus it is that curve that is used to form the surface.

6. Designating a Curve as the Axis Curve of a Stack

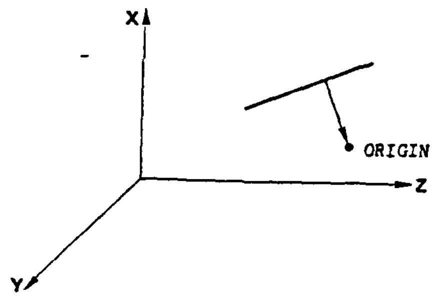
The generation of a surface by stacking space curves along an axis by the operation ITEM="STACK" (Section I-D5) uses an existing curve as the axis. This curve must be so designated by the operation

ITEM = "AXIS"

before the stacking operation is done. The storage location of the existing curve must be given by COREIN or FILEIN (Section I-B3) unless it was generated by the operation immediately preceding the "AXIS" operation.

If the axis is a straight line, a principal normal must be defined by including

ORIGIN = three Cartesian coordinates of a point off the line



The distance of this point from the line is irrelevant.

7. Designating a Curve as an Edge Curve for a Transfinite Interpolation Surface

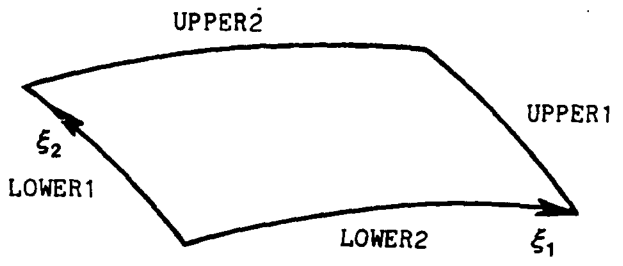
The generation of a surface by transfinite interpolation (Section I-D7) requires four existing curves to be designated to form the four edges of the surface. These four existing curves are designated for this purpose by four usages of the operation

ITEM = "EDGECUR"

The particular edge that the curve is to form is indicated by

EDGE = "edge"

where the value in quotes is "LOWER1", "UPPER1", "LOWER2", or "UPPER2" as follows:



The storage location of the existing curve must be given by COREIN or FILEIN (Section I-B3) unless it was generated by the operation immediately preceding the "EDGECUR" operation.

8. Setting a Corner for a Tensor-Product Surface

The generation of a surface by bi-cubic interpolation from four corners (Section I-D8) requires the setting of the four corner points and the slope vectors at these points. These four corners are set for this purpose by four usages of the operation

ITEM = "SETCOR"

The corner is indicated by

POINT = two indices of corner

The point is set by

R = three Cartesian coordinates of point

and the slope vectors in the two directions on the surface are given as

T1 = three direction cosines of vector in first direction

T2 = three direction cosines of vector in second direction

The spacing in the two directions are given as

SPACE = spacing in first direction, spacing in second

9. <u>Construction of a General Tab Curve</u>

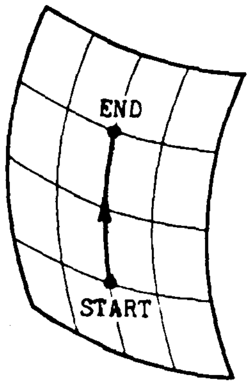
The operation

ITEM = "GENTAB"

selects a portion of a grid line on an existing surface as a tab curve, and sets a slope vector at each point thereon to either a normal or tangent to the surface, for use by the operation ITEM="PATCH" as an end curve of a cubic surface connecting this and another existing surface (Section I-D3). If this surface was generated by the preceding operation, it will automatically be available here. Otherwise, its storage location must be included as COREIN or FILEIN (Section I-B3). The portion of the grid line to be selected is identified by the indices of its end points, given by

START = two indices of first point

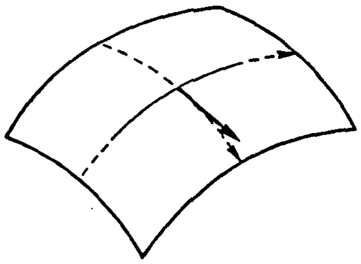
END = two indices of last point



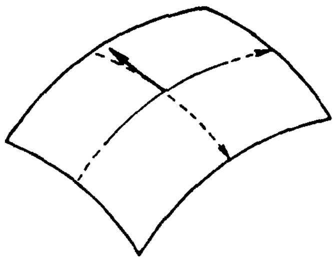
One entry of START must, of course, equal the corresponding entry of END.

The type of slope vector intended is given by one of the following:

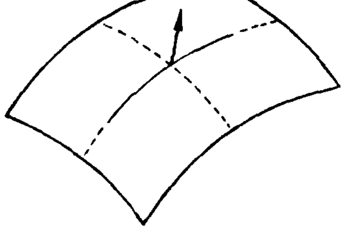
TABTYP = "TANPOS" - positive tangent to crossing lines



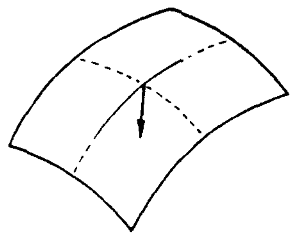
TABTYP = "TANNEG" - negative tangent



TABTYP = "NORPOS" - positive normal unit to surface



TABTYP = "NORNEG" - negative unit normal



Slope vectors based on normals are unit vectors, while those based on tangents reflect the spacing on the surface from which the vector was taken.

10. <u>Construction of an Edge Tab Curve</u>

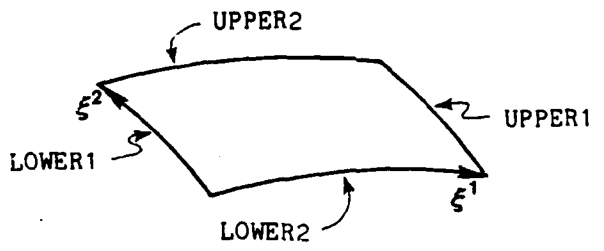
The operation

ITEM = "EDGETAB"

selects an entire edge of an existing surface as a tab curve, and sets a slope vector at each point thereon to either a normal or tangent to the surface, for use by the operation ITEM="PATCH" as an end curve of a cubic surface connecting this and another existing surface (Section I-D3). If this surface was generated by the preceding operation, it will automatically be available here. Otherwise, its storage location must be included as COREIN or FILEIN (Section I-B3). The edge to be selected is identified by

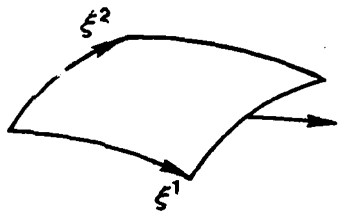
EDGE = "edge"

where one of the following appears in the quotes:

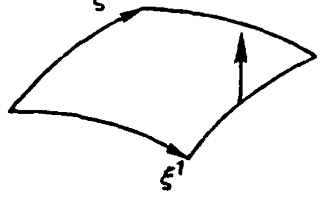


The type of slope vector intended is given as one of the following:

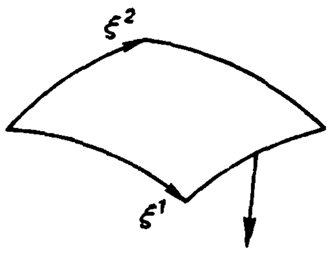
TABTYP = "TANGEN" - outward tangent to surface



TABTYP = "NORPOS" - positive unit normal to surface



TABTYP = "NORNEG" - negative unit normal to surface



Slope vectors based on normals are unit vectors, while those based on tangents reflect the spacing on the surface from which the vector was taken.

11. Construction of a Tab Point

The operation

ITEM = "CURTAB"

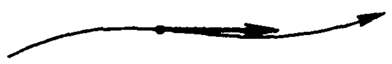
selects a point on an existing curve as a tab point, and sets a slope vector there to either a principal, normal, a bi-normal, or a tangent to the curve, for use by the operation ITEM="PATCH" as an end point of a cubic curve connecting a pair of points on the existing curve (Section I-C3). If this curve was generated by the preceding operation, it will automatically be available here. Otherwise, its storage location must be included as COREIN or FILEIN (Section I-B3). The point to be selected is identified by

POINT = point number

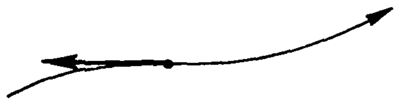
If POINT="FIRST", the first point on the existing curve will be taken (same effect as POINT=1), and if POINT="LAST" the last point will be taken. This eliminates the necessity of ascertaining how many points are on the existing curve in the latter case.

The type of slope vector intended is given by one of the following:

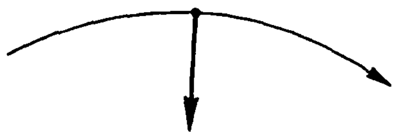
TABTYP = "TANPOS" - positive tangent



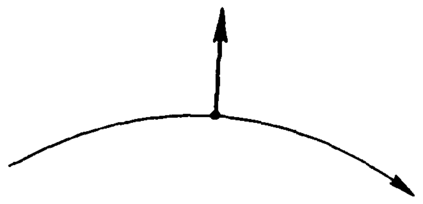
TABTYP = "TANNEG" - negative tangent



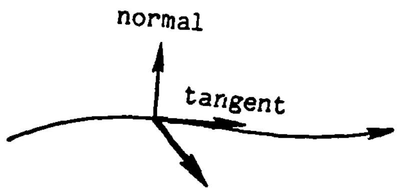
TABTYP = "NORPOS" - positive unit principal normal



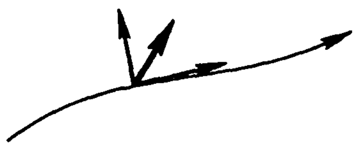
TABTYP = "NORNEG" - negative unit principal normal



TABTYP = "BINPOS" - positive unit bi-normal



TABTYP = "BINNEG" - negative unit bi-normal



12. Direct Setting of a Tab Point

The operation

ITEM = "SETTAB"

directly sets a point and a slope vector for use by the operation ITEM="PATCH" as an end point of a cubic curve (Section I-C3). The point and slope vector are set, respectively, by

R1 = three Cartesian coordinates of point

T1 = three direction cosines of slope vector

The slope vector set in this manner is a unit vector. The spacing on this slope vector will be set in operation "PATCH".

13. Designation of an End Point

The operation

ITEM = "GETEND"

selects a point on an existing surface for use as an end point of a curve to be generated operation ITEM="SCURVE" (Section I-C2), or a line by the operation ITEM="LINE" (Section I-C4). The storage location of the existing surface must be given by COREIN or FILEIN (Section I-B3) unless it was generated by the operation immediately preceding the "GET-END" operation. The point to be selected is identified by

POINT = point number

If POINT="FIRST", the first point on the existing curve will be taken (same effect as POINT=1), and if POINT="LAST" the last point will be taken. This eliminates the necessity of ascertaining how many points are on the existing curve in the latter case.

Whether the selected point is to be the first or second end point of the curve to be generated later is specified by

END = "FIRST" or "SECOND"

14. Designation of an End Point and Slope Vector

The operations ITEM = "GENTAB" (Section I-E9) and "CURTAB" (Section I-E11) serve also to select a point and an attendant unit normal or unit tangent on an existing surface or curve, respectively, for use as an end point and slope vector of a curve to be generated by the operation ITEM="SCURVE" (Section I-C2), or as a corner to be set by operation ITEM="SETCOR" for a tensor product surface (Section I-E8). This mode for these two operations is set by including

END = "end"

where the accepted values are "FIRST" and "SECOND", indicating the end of the curve for which the point is intended. The location of the point on the existing surface(curve) is given with "GENTAB" by

POINT = two indices of point

The second index is omitted with "CURTAB".

15. Splining a Surface(Curve)

The operation

ITEM = "SPLINE"

constructs a bi-cubic surface spline for an existing surface. If the surface(curve) was not generated by the preceding operation, its storage location must be given by including COREIN or FILEIN (Section I-B3). Only one surface(curve) can be splined at a time.

If the surface was the intersected surface of the intersection operation, ITEM="INTSEC", and no later surface has been splined, it is not necessary to invoke ITEM="SPLINE" for that surface.

16. <u>Placing a Surface(Curve) in Current Position</u>

A surface(curve) that has just been generated or treated by an operation is in "current position" and can be treated by the next operation without being stored and retrieved. The operation

ITEM = "CURRENT"

retrieves a surface(curve) from storage and places it in current position. The storage location is given by COREIN or FILEIN (Section I-B3).

17. <u>Storing a Surface(Curve)</u>

Storing a surface(curve) in core or on file may be done in all operations where such is logical by including COREOUT or FILEOUT (Section I-B2). A surface(curve) can also be stored explicitly at any time by the operation

ITEM = "OUTPUT"

with COREOUT or FILEOUT included.

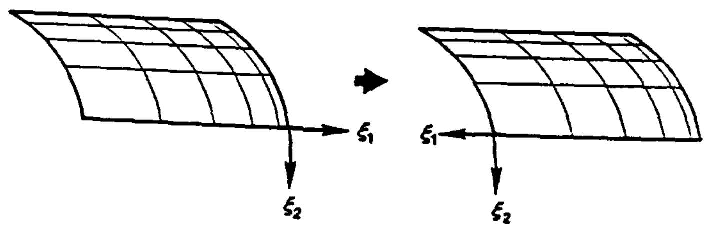
18. <u>Reversing Point Progression and/or Surface Dimensions</u>

The operation

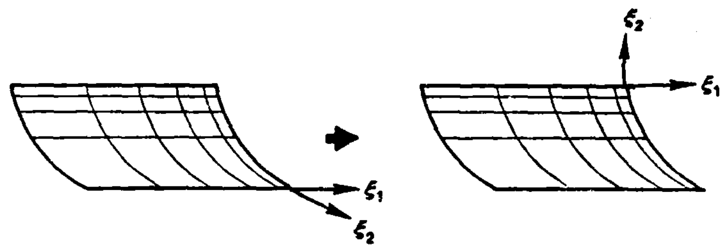
ITEM = "SWITCH"

performs any, or all, of the following actions as specified by the entries given for REORDER:

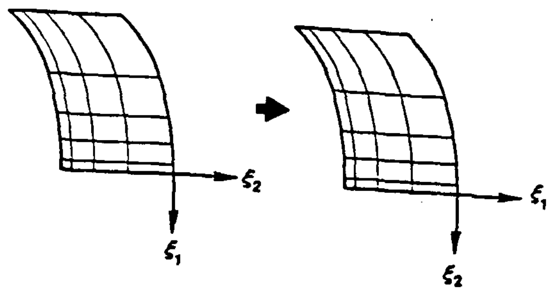
"REVERSE1" - reverses the point progression on a curve, or on all the grid lines in the first coordinate direction on a surface:



"REVERSE2" - reverses the point progression on all of the grid lines in the second coordinate direction on a surface:



"SWITCH" - switches the dimensions of a surface, i.e., the second coordinate direction becomes the first, etc.:



Any one, two, or all three, of these operations (in any order) can be given as the entries for REORDER. In any case the reversals are done before the switching.

The default for REORDER is ("SWITCH", 0,0), simply interchanging the faster and slower running directions. (The "first" coordinate direction is the faster running direction.)

If the surface was not generated or treated by the previous operation, its storage location must be given by including COREIN or FILEIN (Section I-B3). The processed surface can be stored by including COREOUT or FILEOUT (Section I-B2), and can be printed and/or plotted by including OUT and the associated output parameters (Section I-B5).

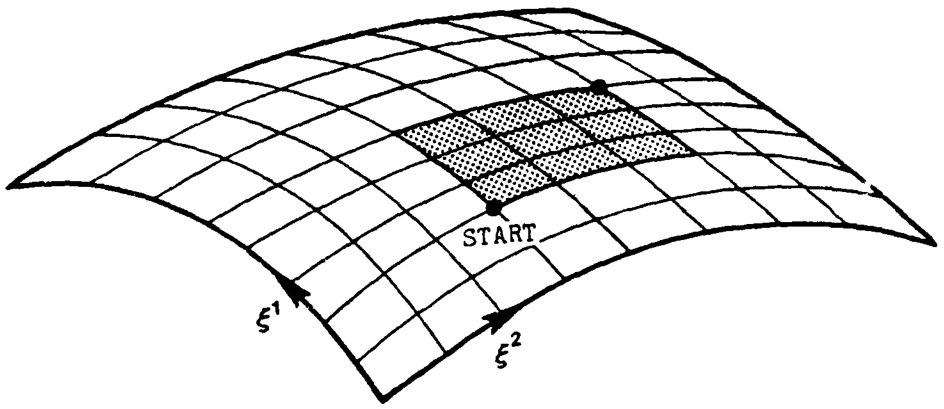
19. Concatenation of Surfaces(Curves)

Surfaces(curves) can be attached to other surfaces(curves) by the operation

ITEM = "INSERT"

Here a surface(curve), with its storage location given by COREIN or FILEIN (Section I-B3), is attached to the surface(curve) that is in current position. The first corner of the former surface is placed at the point on the latter given by

START = first coordinate, second coordinate



(with the second entry omitted for curves). Note that the attachment is made without regard to overwriting or the leaving of gaps.

With curves, START is defaulted to the end of the curve, and can thus be omitted if one curve is to be added to the end of another.

The composite surface(curve) can be stored by including COREOUT or FILEOUT (Section I-B2), and can be printed and/or plotted by including OUT and the associated output parameters (Section I-B5).

20. Duplication of a Surface(Curve) Segment

A segment of an existing surface(curve) can be duplicated to stand alone as a new surface(curve) by the operation

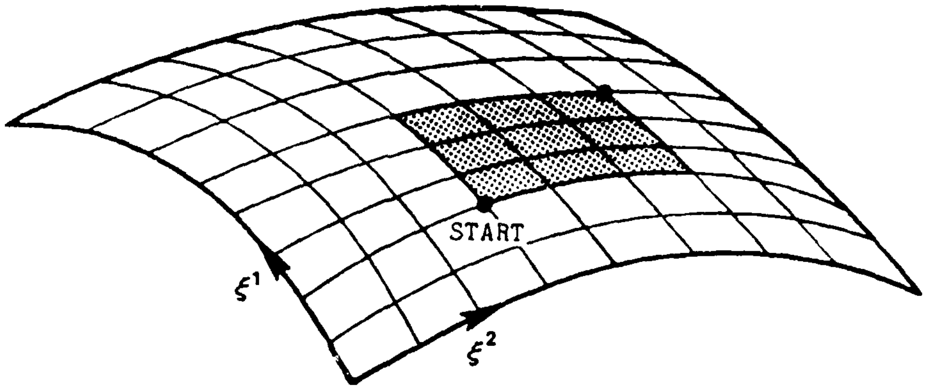
ITEM = "EXTRACT"

The storage location of the existing surface is given by COREIN or FILEIN (Section I-B3). The dimensions of the segment are given by

POINTS = number of points in first direction, number in second (with the second entry omitted for curves) and the location of the first corner of the segment on the larger surface is given by

START = first coordinate, second

(again with the second entry omitted for curves).



The segment can be stored by including COREOUT or FILEOUT (Section I-B2), and can be printed by including OUT and the associated output parameters (Section I-B5).

21. Combining Cores and/or Files onto One Core or File

The operation

ITEM = "COMBINE"

combines several cores and files onto a single core or file. The cores and files to be combined are given by one of the following:

COREIN = first core, second core,---, last core

or

FILEIN = first file, second file,---, last file

A negative entry here implies all cores or files in consequetive order from the preceding entry (which must be positive) to the magnitude of the negative entry, i.e., 3, -6 implies 3, 4, 5, 6. The combined cores and files are stored on the single core or file designated by

COREOUT = combined core

or

FILEOUT = combined file

This combination operation writes the data from the cores and files to be combined point-by-point onto the single output core or file, with no separation between the data from the separate sources. This feature is intended primarily as a convenience in transfering boundary data to the grid code. The combination operation is thus normally used at the end of all the operations used to construct the boundaries, so that only one file need be retained to be read by the grid code. The various surfaces(curves) on this file must, of course, be read by the grid code in the sequence in which they were written on the file, without rewinds.

The operation ITEM="COMBINE" will store a table of contents of the combined file if CONTENT="YES". This table consists of the COREOUT number and the surface(CURVE) dimensions for each segment on the file. This table can be read by the grid code and used there to set up the boundary configuration (cf. Section I-C23 of Volume III).

Alternatively, if HEAD="YES" is included, the dimensions of each segment, preceded by the counter, will be placed on a single line before each segment on the file (Section I-B5). This is useful for the construction of a file of segments for plotting.

22. Copying Cores or Files

The operation

ITEM = "COPY"

copies one or more cores or files onto other cores or files. Cores can be copied to files, or vice versa. The cores or files to be copied are given by one of the following:

COREIN = first core, second core,---, last core

or

FILEIN = first file, second file,---, last file

A negative entry here implies all cores or files in consequetive order from the preceding entry (which must be positive) to the magnitude of the negative entry (Section I-E21). The cores or files for the copies are given by one of the following:

COREOUT = first core, second core,---, last core

or

FILEOUT = first file, second file,---, last file

Again negative entries may be used as above.

23. Setting an Input Value as a Sum or Product

The operation

ITEM = "SETVAL"

calculates an integer value as a sum or product and stores the value for later use as an input value in another operation. The type of calculation is indicated by

MATH = "type of calculation"

where the possibilities are the following:

"SUM" - sum of values

"DIF" - difference of two values

"PRODUCT" - product of values

The integer values in the calculation are given as

TERMS = value, value, value, ---

and the storage location by

VALOUT = location

where the location is a positive integer. Negative entries of TERMS refer to values stored by previous usage of "SETVAL". The maximum number of values that can be included in the calculation in NVALMX, and the maximum storage location in DVAL, both of which can be changed by <u>global</u> edits. The storage here has nothing to do with that for surfaces (curves).

The operation "SETNUM" sets integer values as in the grid code, Section I-C1, Vol. III.

24. <u>Projection onto a plane</u>

The operation ITEM="SCALE" (Section I-E3) with a zero entry in SCALE, and unity for the other two values, will have the effect of projecting a surface(curve) onto the plane corresponding to the zero entry.

25. <u>Cartesian coordinates at numbered points</u>

The operation

ITEM="POINT"

sets the Cartesian coordinates, given in R(3), of a numbered point indicated by the positive integer given for POINT (cf. Section I-B9):

```
$INPUT ITEM = "POINT",  POINT = __,  R = __,__,__ $
```

Note that R can be set by a previous use of "GETEND" (Section I-E13), and is omitted in that case.

26. <u>Surface parametric coordinates of a surface(curve)</u>

The operation

ITEM="CORPAR"

generates the values of the surface parametric coordinates for each point on an existing surface(curve) constructed on a curved surface. The curved surface must have been splined by operation ITEM="SPLINE" (Section I-E15), of course. The parametric values can be stored by including COREOUT or FILEOUT (Section I-B2), and can be printed by including OUT="PRINT" (Section I-B5). This operation is useful in generating boundary segments (i.e., values of the two parametric coordinates at each point on a line of the curved surface) for input to the grid code (Section I-C24 of Vol. III).

27. Distributing Points on a Surface with Specified Spacing

Distribution of points on an existing surface with the relative or actual arc-length spacing specified at one or both edges in each direction, or at a specified point between the edges, is done by the operation

ITEM = "SURDIST"

The number of points to be placed on the surface is set by

POINTS = number in first direction, number in second direction

and the distribution is set by DISTYP and the associated distribution parameters (Section I-B6). The first entry (or first pair of SPACE) of each of these refers to the first direction on the surface, etc. The spacings given will be taken as actual arc lengths if

RELATIV = "NO"

is included. The two entries in RELATIV here refer to the two directions on the surface. It is not necessary to include the total arc length with relative spacings here since it is known from the input surface. The spacing can be set in several ways as discussed in Section I-B6.

The surface on which the points are to be placed must have been generated prior to this operation. If this surface was generated by the preceding operation, it will automatically be available here. Otherwise, its storage location must be given by including COREIN or FILEIN (Section I-B3). Finally, if the point distribution is to be stored, the storage location must be given (Section I-B2), and printing and/or plotting can be called for (Section I-B5).

F. OTHER OPERATIONS

1. Relative Distribution

The setting of a set of relative distribution factors (monotonic on 0-1) is involved in many operations, and is normally done in that connection by including DISTYP, or its equivalent, and the associated distribution parameters (Section I-B6) on the input statement for the operation. A distribution can, however, be set separately by the operation

ITEM = "DISTRIB"

with DISTYP, etc., included. The number of points in the distribution is given as

POINTS = number

Here if actual arc lengths are given as the spacing (with RELATIV="NO"), it is necessary to also include the total arc length as

TOTARC = total arc length

The N-point distribution is placed in the array DISTRIB(N,1).

2. Arc Length on a Curve

The arc length (actually chord length) at each point on an existing curve can be calculated by the operation

ITEM = "ARC"

The number of points on the curve is given as

POINTS = number

If this curve was generated by the preceding operation, it will automatically be available here. Otherwise, its storage location must be included as COREIN or FILEIN (Section I-B3). The arc lengths at the N points are placed in the array ARCLEN(N).

3. Unit Tangent and Principal Normal, Curvature and Arc Length on a Curve

The unit tangent and principal normal, curvature, and arc length at each point on an existing curve can be calculated by the operation

ITEM = "CURVEC"

The number of points on the curve is given as

POINTS = number

If this curve was generated by the preceding operation, it will automatically be available here. Otherwise, its storage location must be included as COREIN or FILEIN (Section I-B3). If the curve is a straight line it is necessary to include

ORIGIN = three Cartesian coordinates

to specify a point off the line to define the principal normal. The tangent, normal, curvature, and arc length at the N points are placed in the arrays TANGEN1(3,N,1), NORMAL(3,N,1), CURVAT(N), and ARCLEN(N).

4. Unit Tangents and Normal on a Surface

The two unit tangents and the unit normal at each point on an existing surface can be calculated by the operation

ITEM = "SURVEC"

The number of points in the two directions on the surface are given by

POINTS = number in first direction, number in second

If this surface was generated by the preceding operation, it will automatically be available. Otherwise, its storage location must be given as COREIN or FILEIN (Section I-B3). The tangents and normal at the N1xN2 points are placed in the arrays TANGEN1(3,N1,N2), TANGEN2(3,N1,N2), and NORMAL(3,N1,N2).

G. ERROR MESSAGES

ALL FOUR CORNERS ARE REQUIRED

This occurs when not all four corners of a tensor-product surface (Section I-D8) have been designated. Four "SETCOR" operations must precede the "TENSUR" operation.

ALL FOUR EDGES ARE REQUIRED

This occurs when not all four edges of a transfinite interpolation surface (Section I-D7) have been designated. Four "EDGECUR" operations must precede the "TRANSUR" operation.

ANGLES APPEAR TO BE IN RADIANS

This warning is given when a non-zero value less than 2π in magnitude is given for ANGLE, because values for ANGLE must be given in degrees. This is only a warning, since a small angle may be actually intended.

<ANGPTS> IS REQUIRED

This occurs when the number of rotation angles for the generation of a surface by rotation of a curve (Section I-D4) is not specified.

AT LEAST 3 POINTS REQUIRED ON CONNECTING CURVES

At least 3 points must be called for in CURVES on the connecting curves between two tab curves (Section I-D3,C3).

AT LEAST 5 POINTS NECESSARY FOR SPLINE

A curve must have at least 5 points in order to be splined (Section I-E15).

AT LEAST 2 ROTATION POSITIONS REQUIRED
AT LEAST 2 STACKING POSITIONS REQUIRED
AT LEAST 2 BLENDING POSITIONS REQUIRED

This occurs when only one position is prescribed for a surface to be created by rotation, stacking or blending curves (Section I-D4,D5, D6). This would not create a surface.

<AXCOS> OR <COSINES> MUST BE A UNIT VECTOR

This occurs when an axis tangent (AXCOS), or the z-axis of a rotated system(COSINES), is not given as a unit vector.

<AXCOS & NORCOS> CANNOT BE THE SAME

This occurs when the same unit vector is given for both the axis tangent and principal normal when a surface is generated by stacking curves (Section I-D5).

BAD FIRST CORE IN GROUP
BAD FIRST FILE IN GROUP

This occurs when multiple entries (Section I-B2,B3) are given for COREIN or COREOUT (or for FILEIN or FILEOUT), and the first entry is negative. A negative entry for a later entry indicates all cores(files) from the preceding entry, i.e. 5,-9 means 5,6,7,8,9. The first entry, however, must be positive.

BAD VALUE FOR ---

This occurs when an unrecognized value is given for the quantity indicated. The acceptable values are indicated below.

OUT	:	"PRINT, "PLOT"
FRAME	:	"OLD", "NEW"
REWOUT	:	"YES", "NO"
TRIAD	:	"YES", "NO"
ITRIAD	:	"YES", "NO"
FORM	:	"UNFORM", "LIST", "E"
IFORM	:	"UNFORM", "LIST", "E"
REWIN	:	"YES", "NO"
CHANGE	:	"YES", "NO"
SIZE	:	positive real
SYMBOL	:	-1,0,1
TOTARC	:	positive real
RELATIV	:	"YES", "NO"
WIDTH	:	positive real
SEMIAX	:	positive real
LENGTH	:	positive real
ANGLE	:	$\lvert ANGLE(2) - ANGLE(1)\rvert \leq 360$
ECCENT	:	non-negative real
SYMTOT	:	positive real
END	:	"FIRST", "LAST", integer
MATH	:	"SUM", "SUM+1", "DIF", "DIF-1", "PRODUCT"
EDGE	:	"LOWER1", UPPER1", "LOWER2", "UPPER2"

TYPE	:	"CIRCLE", "CIRARC", "ELLIPSE", "ELLIARC", "PARABOL", "HYPERBOL", "SPHERE", "SPHSEG", "ELIPSOID", "ELLISEG", "ELLICONE", "ELLIPAR", "QUAD", "NATURAL"
TABTYP	:	"NORNEG", "NORPOS", "TANNEG", "TANPOS", "BINNEG", "BINPOS"
DISTYP	:	"LINEAR", "BOTH", "TANH", "SINH", "INTERIOR" (and equivalenced)
REORDER	:	"SWITCH", "REVERSE1", "REVERSE2"
VALOUT	:	positive integer

BAD VALUE FOR SPLINE END TYPE

This occurs when an unrecognizable end type is specified for the spline (Section I-E15).

BOUNDING CURVES MUST HAVE SAME NUMBER OF POINTS

This occurs when the two boundary curves set for the generation of a surface by rotation ("ROTATE", Section I-D4), stacking ("STACK", Section I-D5) or blending ("BLEND", Section I-D6) do not have the same number of points.

CORE STORAGE LOCATION EMPTY

This occurs when a surface is asked for (by COREIN) from a core storage location that has not been filled by a previous COREOUT (Section I-B2).

CORE STORAGE NUMBER TOO LARGE

This occurs when a core or file number is used which is larger than the maximum allowed. Increase the indicated dimension parameter glob-ally. The suggested value is adequate only for the present instance.

CORES & FILES CANNOT BE TREATED TOGETHER

This occurs when either both COREIN and FILEIN, or both COREOUT and FILEOUT, are included. It is not possible to use input from, or output to, both core and file storage simultaneously. It is possible, however, to use input from one source and output from the other, or vice versa. (Section I-B2,B3).

CORE OVERWRITTEN WITHOUT BEING USED

This warning occurs when a second surface is written into a core storage location (with COREOUT, Section I-B2) without using the surface already stored there.

CORNERS DO NOT MATCH

This occurs when the Cartesian coordinates of the ends of two edges at a corner of a transfinite interpolation surface (Section I-D7) are not coincident.

CORNERS MUST FORM A RECTANGLE

This occurs when the indices of the four corners set for a tensor-product surface (Section I-D8) do not form a rectangle in computational space.

<COSINES>

This occurs when the new y-axis of a rotated system is not given as a unit vector.

<CURPTS> IS REQUIRED

This occurs when the number of points on a curve is not specified either directly or indirectly (Section I-B9).

<CURPTS & RADPTS> ARE REQUIRED

This occurs when the number of points in either the radial or angular direction on a flat surface bounded by a conic-section curve (Section I-D1) is not specified.

CURVE IS A POINT

This will occur when both end points of a cubic curve (Section I-C2) are coincident in space (Use the "LINE" operation to deliberately produce a degenerate curve). It will also occur when the number of points on a curve is 1, but a curve is required.

CURVE IS A SURFACE

This occurs when the curve segment identified by START and END (Section I-E9) is not a curve but a surface. Two entries of END must be the same as the corresponding two entries of START.

CURVE IS OFF THE SURFACE

This occurs when the curve identified by START and END (Section I-E9) does not lie entirely on the surface, i.e., some index of START or END is not within the dimensions of the surface.

<CURVES> IS REQUIRED

This occurs when the number of curves to be on a surface generated by blending two curves (Section I-D6) is not specified.

CURVE MUST HAVE AT LEAST 3 POINTS

At least three points are necessary on the curve.

CURVE MUST HAVE AT LEAST 4 POINTS

Specified spacing on both ends of a curve requires at least four points on the curve (Section I-C1).

DEGENERATE CURVE

Two successive points on the curve are coincident.

DEGENERATE CONNECTING CURVES

This occurs when a patch surface is generated connecting two tab curves (Section I-D3,C3), and the tab curves are coincident.

DIMENSIONS EXCEEDED : N1,N2 = ---

This occurs when the dimensions of a surface exceeds the maximum allowed. Increase the dimension parameters DIM1 and DIM2 globally. The suggested values are adequate only for the present instance.

<EDGE> IS REQUIRED

This occurs when the particular edge intended for a transfinite interpolation surface (Section I-D7) is not specified.

EDGES DO NOT MATCH

This occurs when two opposite edges set for a transfinite interpolation surface (Section I-D7) do not have the same number of points.

<END> IS REQUIRED

This will occur when a spacing is being set from another curve (Section I-D10), but the particular end ("FIRST" or "LAST") of the other curve from which the spacing is to be taken is not indicated. It will also occur with the "GETEND" operation (Section I-E13) when the end ("FIRST" or "LAST") for which the selected point is to be used is not indicated.

EXTRACTION IS OFF THE SURFACE

This occurs when the section indicated for extraction is not contained on the surface. Check START and POINTS.

FEMALE SURFACE LOCATION REQUIRED

This occurs when no female surface is indicated for an intersection (Section I-C5).

FEMALE SURFACE MUST BE AT LEAST 5X5

The female surface must be at least 5x5 in order to be splined (Section I-C5).

FEWER OUTPUT CORES THAN INPUT CORES
FEWER OUTPUT FILES THAN INPUT FILES

This occurs when cores or files are being copied (Section I-E22) and the total number of copies indicated (COREOUT or FILEOUT) is less than the total number of originals indicated (COREIN or FILEIN).

FILE OVERWRITTEN WITHOUT BEING USED

This occurs when a file is overwritten (by FILEOUT, Section I-B2) being used. This will occur harmlessly with the "COMBINE" operation.

FILE STORAGE NUMBER TOO LARGE

See CORE STORAGE NUMBER TOO LARGE.

FIRST POINT NOT SET

This occurs when a point number, instead of the three Cartesian coordinates, is given for R1 (the first point on a curve), but no coordinates have been previously attached to that point number by "POINT" operation (indirect addressing, Section I-B9).

IDENTICAL BOUNDING CURVES

It is necessary to designate two bounding curves when a surface is to be created by blending (Section I-D6).

IDENTICAL TAB CURVES

Two tab curves must be designated before the "PATCH" operation (Section I-D3,C3) is invoked.

INNER ITERATION DOES NOT CONVERGE

This occurs when the Newton iteration along the curves of the male surface does not converge in determining an intersection (Section I-C5). Check to see if the ξ^2 curves on the male surface really can intersect the female surface. If it is the ξ^1 curves that intersect, use an ITEM="SWITCH" on the male surface before the "INTSEC" operation.

<ITERMS> IS REQUIRED

This occurs when no terms for the calculations are given (Section I-E23).

LAST POINT NOT SET

See FIRST POINT NOT SET. Here R2 (the last point on a curve) is in question.

LATITUDE MUST BE POSITIVE

This occurs when a negative value is given for a latitude angle defining a conic-section surface (Section I-D2). Latitude angles must range on 0-180.

LINEAR DISTRIBUTION WITH <DISTYP> NOT LINEAR

This occurs when the spacing specified is such as to produce a linear distribution, but a nonlinear distribution has been calleo for (Section I-B6). Change DISTYP (or equivalenced quantity) to "LINEAR".

<LONPTS & LATPTS> IS REQUIRED

This occurs when the number of points in either the longitudinal or latitudinal direction on a conic-section surface (Section I-D2) is not specified.

MALE SURFACE MUST BE AT LEAST 1X5

The male surface must be at least 1x5 in order to be splined along the ξ_2 curves (Section I-C5).

<MATH> IS REQUIRED

This occurs when the type of calculation has not been given (Section I-E23).

NEGATIVE OR ZERO RESULT

This occurs when the calculation for a number of points to be stored (Section I-E23) produces a non-positive number.

NEWTON ITERATION DOES NOT CONVERGE

This occurs when the Newton iteration for the surface parametric coordinates from the Cartesian coordinates does not converge. Probable cause is a strangely shaped surface.

NO AXIS

This occurs when no axis curve has been designated from the generation of a surface of stacking curves (Section I-D5).

NO BOUNDING CURVE SET

This occurs when no bounding curve for the generation of a surface by rotation ("ROTATE", Section I-D4), stacking ("STACK", Section I-D5) has been set.

NO OUTPUT FILE OR CORE

This occurs when an output file (FILEOUT) or core (COREOUT) is expected but none is given (Section I-B2).

NO STORED VALUE FOR ---

This occurs when a negative value, indicating by magnitude the storage location of a value previously stored by a "SETNUM" or "SETVAL" operation (Section I-E23), is given for the noted quantity, but no value has actually been put in the location indicated.

NO STORED VALUE IN LOCATION

This occurs when a term of the calculation is indicated by a negative number to have been previously stored by a "SETNUM" or "SETVAL" operation (Section I-E23), but no value has actually been stored in that location.

NO SURFACE

This occurs when a surface(curve) is being generated on a curved surface, but the surface has not yet been splined (Section I-B10).

<NORCOS>, OR <COSINES>, MUST BE A UNIT VECTOR

This occurs when a normal vector (NORCOS), or the new x-axis of a rotated system (COSINES), is not given as a unit vector.

NOT A SURFACE

This occurs when one of the dimensions of the surface is given as 1, but a surface is required.

NOT ENOUGH POINTS GIVEN

This occurs when the total number of triads (diads in 2D) of Cartesian coordinate values given in VALUES when reading points from the namelist (Section I-E16) is less than the number indicated on the surface by POINTS.

OUTER ITERATION DOES NOT CONVERGE

This occurs when the Newton iteration on the female surface does not converge in determining an intersection (Section I-C5). See also the notes on the inner iteration not converging.

POINT IS OFF THE CURVE

This occurs when the point selected is off the curve(surface), i.e., the point indices exceed the surface dimensions.

<POINT> IS REQUIRED

This will occur when the number intended for a point is not given with the "POINT" operation (Section I-E25). It will also occur with the "GETEND" (Section I-E13) or "CURTAB" (Section I-E11) operations when the point to be selected is from the curve(surface) is not indicated. It will also occur with the "SETCOR" operation (Section I-E8) when the indices for the surface corner are not given.

<u><POINTS> IS REQUIRED</u>

This will occur when the number of points on a curve, or the dimension of a surface, is not indicated either directly or indirectly (Section I-B9). It will also occur when a surface is being read from a file (by FILEIN, Section I-B3), or from the namelist (with VALUES, Section I-E16), and the dimensions of the surface are not given.

<u><R1 & R2> ARE REQUIRED</u>

This occurs when either of the end points of a line (Section I-C4) is not specified.

<u><R & T> ARE REQUIRED</u>

This occurs when either the point, or the unit tangent, for the end of a cubic curve (Section I-E14) or a tab point (Section I-E12) is not specified.

<u><R1 & T1> ARE REQUIRED</u>
<u><R2 & T2> ARE REQUIRED</u>

This occurs when either the point, or the unit slope vector, for an end point of a cubic (Section I-C2) is not specified.

<u><R> IS REQUIRED</u>

This occurs when no Cartesian coordinates are specified for the point (Section I-E25).

<R,T1 & T2> ARE REQUIRED

This occurs when either the point, or one of the two unit slope vectors, for the corner of the tensor-product surface (Section I-E8) is not specified.

<RADIUS> IS REQUIRED

This occurs when the radius of a circle or sphere is not specified.

<SCALE> IS REQUIRED

This occurs when no scale factors are given (Section I-E3). (Values intended as unit do not have to be included.)

<SEMIAX> IS REQUIRED

This occurs when the semi-axes of an ellipse or ellipsoid are not specified.

<SNUMER> IS REQUIRED

This occurs when a scale factor is being calculated as a ratio, but no numeration is given.

<SPACE> IS REQUIRED

This occurs when either of the two spacings at a corner of the tensor-product surface (Section I-D8) is not specified.

<SPACE> IS REQUIRED WITH NORMAL TABS

When normal tabs are used for a patch surface (Section I-D3), the spacing on the tab must be specified.

SPACING & LOCATION REQUIRED

Both the spacing(SPACE) and the arc-length location(ARCINT) must be specified when an interior spacing is set (Section I-B6).

SPACING EXCEEDS TOTAL ARC LENGTH

This will occur when an absolute spacing is given, i.e., with RELATIV="NO", that exceeds the total arc length of the curve. It will also occur in like manner with the "DISTRIB" (Section I-B6) operation when the spacing exceeds the specified total arc length, TOTARC.

<START & END> ARE REQUIRED

This occurs when either of the two points that identify the curve segment (Section I-E9) is not specified.

<START> IS REQUIRED

This occurs when the starting point for insertion (Section I-E19) or extraction (Section I-E20) is not specified.

SURFACE IS A CURVE

This occurs when one dimension of the surface is 1, but a surface is required.

SURFACE MUST BE AT LEAST 3X3

The surface must have at least 3 points in each direction.

SYSTEM RESTRICTED FILE NUMBER

Certain file numbers are reserved for system usage. These restrictions will be peculiar to the particular installation, and the checking statement in subroutine PUT can be changed to fit the local restrictions.

<T1>, OR <T>,MUST BE A UNIT VECTOR
<T2> MUST BE A UNIT VECTOR

This occurs when a slope vector is not given as a unit vector.

TAB & BOUNDING CURVE OPERATIONS CANNOT BE MIXED

This occurs when a tab curve and a bounding curve have both been designated. An operation involving tab curves must be completed before another operation involving bounding curves can be begun, and vice versa.

TAB CURVE MUST HAVE AT LEAST 3 POINTS

This occurs when an edge of a surface that has less than three points is designated as a tab curve (Section I-E10) for a patch surface.

TAB CURVES MUST HAVE SAME NUMBER OF POINTS

This occurs when the two tab curves (Section I-D3) set do not have the same number of points.

<TABTYP & EDGE> ARE REQUIRED

This occurs when either TABTYP or EDGE is omitted. The type of tab must be set by the former to be a normal or tangent to the surface, and an edge of the surface must be designated by the latter to be the tab curve.

<TABTYP> IS REQUIRED

This occurs when no type of the tab is given. The tab must be designated to be either a tangent or normal to the surface.

<TERMS> IS REQUIRED

This occurs when no terms for the calculation are given (Section I-C23).

TOO MANY BOUNDING CURVES

This occurs when more than two bounding curves have been set for a surface generated by rotation ("ROTATE", Section I-D4), stacking ("STACK", Section I-D5). The operation must be invoked after two bounding curves are set.

TOO MANY CORES AT ONCE ---
TOO MANY FILES AT ONCE ---

This occurs when the total number of cores or files given exceeds the maximum allowed (Section I-B2,B3). Increase the indicated dimension parameter globally. The suggested value is adequate only for the present instances.

TOO MANY CORNERS

This occurs when more than four corners are set for a tensor-product surface (Section I-D8). The "TENSUR" operation must be invoked after four corners have been set.

TOO MANY EDGE CURVES

This occurs when more than four edge curves for a transfinite interpolation surface (Section I-D7) are set. The operation "TRANSUR" must be invoked after four edge curves have been set.

TOO MANY POINTS FOR CORE STORAGE

This occurs when the dimensions of a surface are too large for the core storage array (Section I-B2). Increase DIMSS globally. (The suggested value is adequate only for the present instance.)

TOO MANY POINTS

This indicates that the point number given exceeds the maximum allowed (Section I-B9). The dimension parameter DPNT must be increased globally. The suggested value is only adequate for the present point number.

TOO MANY TERMS

This indicates that the number of terms allowed has been exceeded (Section I-C23). Increase the dimension parameter NVALMX. The suggested value is adequate only for the present instance.

TOO MANY VALUES STORED

This indicates that the storage locations for calculated values (Section I-C23) have been exhausted. Increase the dimension parameter DVAL. The suggested value is adequate only for the present instance.

TOTAL ARC LENGTH REQUIRED

This occurs when a relative spacing (RELATIV="NO") is indicated, but no total arc length is available. The total arc length must be specified in this case by including TOTARC.

TOTAL ARC LENGTH REQUIRED WITH NORMAL TABS

With relative spacing (RELATIVE="NO") on normal tabs for a patch surface (Section I-D3), a total arc length for the connecting curves must be specified by including TOTARC.

TOTAL ARC LENGTH TOO SMALL

This will occur when relative spacing is used in the generation of a patch surface between two tab curves (with ITEM="PATCH", Section I-D3) and the total arc length specified by TOTARC is less than the straight line distance between the tab curves. It will also occur in like manner of a cubic curve (ITEM="SCURVE", (Section I-D2).

<TOTARC> REQUIRED WITH RELATIVE SPACING

This occurs when a distribution function is being generated (Section I-E1) with relative spacing specified, but no total arc case is specified.

TWO BOUNDING CURVES REQUIRED

Two bounding curves must be designated for the generation of a surface by blending curves, i.e., two "BOUNCUR" operations must precede the "BLEND" operation.

TWO TAB CURVES REQUIRED

This occurs when only one tab curve has been designated from a patch surface (Section I-D3. Two tab curve operations ("GENTAB", "EDGETAB", "SETTAB", "CURTAB") must precede the "PATCH" operations.

<TYPE> IS REQUIRED

This will occur when the type of a conic-section curve ("CONICUR") operation, or a conic-section surface ("CONISUR" or "FLATSUR" operation), is not given.

UNKNOWN NAMELIST ITEM

This occurs when an unrecognizable operation is invoked, i.e., for ITEM.

<VALOUT> IS REQUIRED

This occurs when no storage location for the calculated value is specified (Section I-C23).

<WIDTH & LENGTH> ARE REQUIRED

Both the width and length for an open circular or parabolic arc must be specified.

<WIDTH, LENGTH, & ECCENT> ARE REQUIRED

The width, length, and eccentricity of an open elliptical arc must be specified.

<WIDTH, LENGTH, & SYMTOT> ARE REQUIRED

The width, length, and asymptote angle must be specified for a hyperbolic arc.

ZERO ARC LENGTH

Two successive points on the curve are coincident.

ZERO NORMAL

This occurs when neighboring points on a surface are coincident.

PART II - CODE OPERATION

A. PARAMETERS AND VARIABLES

The surface dimension parameters are the integers DIM1 and DIM2, which are the maximum dimensions of a surface that can be handled. The maximum number of points allowed on a curve is DIM1. These are set by identical PARAMETER statements in each routine in which they are involved, and therefore can be changed by global edits.

1. Primary Arrays

The primary arrays, which are transferred between routines by COMMON/RAY/, are the following:

CORD (3,DIM1,DIM2) or CORDI(3,DIM1,DIM2) - three Cartesian coordinates of grid points on a surface. (real) (The array CORDI is used to input a surface to a routine to be changed, with the result being returned in the array CORD.)

NORMAL(3,DIM1,DIM2) - three Cartesian components of the normal to a surface. (real)

TANGEN1(3,DIM1,DIM2) and TANGEN2(3,DIM1,DIM2) - three Cartesian components of the tangents to the two families of grid lines on a surface. (real)

ARCLEN(DIM1) - arc length at each point on a curve. (real)

CURVAT(DIM1) - curvature at each point on a curve. (real)

DISTRIB(DIM1,2) - two sets of relative distribution factors, with monotonic variation on 0-1. (real)

CURV(3,DIM1,4) - three Cartesian coordinates of points on four edge curves from which a surface is interpolated. (real)

TABVEC(3,DIM1,2) - three Cartesian components of slope vectors on two curves between which a surface is to be constructed. (real)

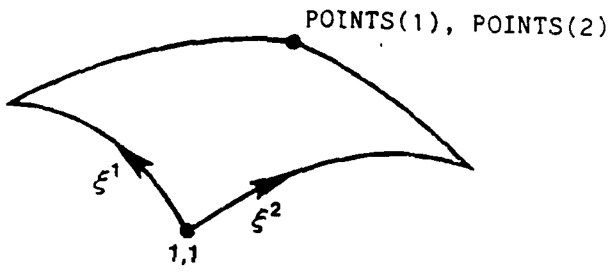
AXIS(3,DIM1) - three Cartesian coordinates of points on a space curve to be used as an axis along which curves are stacked to form a surface. (real)

SCALE(3,DIM1) -	three scale factors for the Cartesian components at each point on a surface. (real)

2. Surface(Curve) Parameters

The surface(curve) parameters, which are transferred between routines through COMMON/PAR/, are as follows:

POINTS (2) or POINTSI(2) -	the two dimensions of a surface. (integer) (The array POINTSI is used in reference to a surface being input to a routine.)



POINTS is equivalenced with N1,N2; POINTSI with NI1, NI2; POINTS(1) with N; and POINTSI(1) with NI. The following equivalences are also made (all integer):

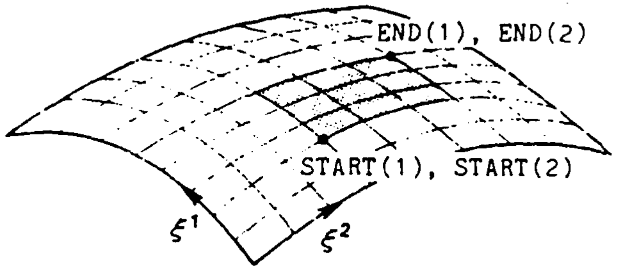
<u>POINTS(1)</u>

LONPTS	:	number of longitude points
CURPTS	:	number of points on a curve

<u>POINTS(2)</u>

LATPTS	:	number of latitude points
ANGPTS	:	number of angular points
RADPTS	:	number of radial points
TABPTS	:	number of points on a tab curve
AXPTS	:	number of points on the axis of a stack
CURVES	:	number of joining curves

START(2),END(2) - the two curvilinear coordinates of each of two opposite corners defining surface segment. (integer)



START is equivalenced with POINT, the two curvilinear coordinates of a point to be selected from a surface, and with REORDER, the indicator for coordinate switching and reversal.

LAT(2),LON(2) - two latitude and longitude angle limits for conic-section surfaces. (real)

SEMIAX(3) - three semi-axes for an ellipsoid. (real) SEMIAX is equivalenced with WIDTH(2) and with RADIUS, also conic section curve parameters.

ORIGIN(3) - three Cartesian components of the new origin in a translation. Also coordinates of point defining a normal to a line. (real)

ANGLE(2) - two angle limits. (real)

R1(3),R2(3) - three Cartesian coordinates of the end points of a curve. (real)

PARAM(3,3) - matrix of nine rotation parameters. (real)

PARAM is equivalenced with COSINES for rotation described by the nine direction cosines, and with EULER for rotation described of the three Euler angles. COSINES, in turn, is equivalenced as follows (all real):

COSINES(1,1) with NORCOS(3) : direction cosines of principal normal to an axis

COSINES(1,3) with AXCOS direction cosines of an axis

TYPE - type of surface(curve). ("---")

LENGTH, ECCENT, SYMTOT - conic-section parameters. (real)

TABTYP - type of tab used to generate a connecting surface. ("---")

EDGE - type of edge to which a connecting surface is attached. ("---")

3. I/O Parameters

The I/O parameter array dimension parameters are the following:

DIMSS - total number of points that can be stored on core. (integer)

DIMV - maximum number of points that can be read for a surface(curve) from the namelist. (integer)

DFIL - maximum number of surfaces(curves) that can be stored on file. (integer)

DCOR - maximum number of surfaces(curves) that can be stored in core. (integer)

DVAL - maximum number of values that can be stored for input as values of quantities on the input statements. (integer)

NVALMX - maximum number of terms that can be involved in the calculation of a stored value. (integer)

DPNT - maximum number of numbered points. (integer)

These are set by identical PARAMETER statements in the main program and in subroutines PUT and GET. From these parameters, the integers DFIL4, DCOR4, and DIMV3 are calculated as the indicated multiples, i.e., DFIL4 = 4*DIFL.

The input/output arrays and parameters, transferred between routines through COMMON/IO/, are the following:

STORE(3,DIMSS) - core storage array. (real)

NSTOR(DCOR) - core storage segment numbers array

LSTOR(DCOR) - core storage location numbers array

FILES(0:4,-2:DFIL) - file parameter array. (integer)

CORES(0:4,DCOR) - core parameter array. (integer)

VALUES(DIMV3) - array for input of surface from namelist. (real)

RMIN(3),RMAX(3),VIEW(3),SIZE(2) - plot parameters. (real)

SYMBOL,NPLOT - plot parameters. (integer)

FRAME - plot parameter. ("---")

OUT(2) - type of output. ("---")

NSTORE,MSTORE - core storage segment number

LSTORE - core storage location number

PSTORE - core storage parameter. (integer)

CORIN,COROUT - core storage numbers. (integer)

FILIN,FILOUT - file storage numbers. (integer)

LABIN,LABOUT - labels for input/output. ("---")

REWIN,REWOUT - file rewind indicators. ("---")

NC1,NC2 - dimensions of current surface. (integer)

FORM,TRIAD,HEAD,NHEAD - file format parameters. ("....")

4. Distribution Parameters

The point distribution parameters, which are transferred between routines in COMMON/DISTRI/ are the following:

DISTYP(2) - type of distribution. ("---")

RELATIV(2) - relative distribution indicator. ("---")

SPACE(2,2) - two spacings for each of two curves. (real)

TOTARC - total arc length. (real)

5. Spline Parameters

The spline parameters, transferred in COMMON/SPLINE/, are

SUR(3,DIM1,DIM2,0:3) - three Cartesian components of surface spline array. (real)

CUR(3,DIM1,0:1) - three Cartesian components of curve spline array. (real)

B. INPUT AND SETUP

The code operates by responding to successive reads of NAMELIST/INPUT/, with the action on each read determined by an alphanumeric value given to the integer ITEM. Defaults are reset after each read. Standard values are set for all input quantities for which a value can be anticipated, and recognizable unreasonable values, e.g. zero for a number of points or a radius, are assigned to input quantities which must be included, so that error checks can be made for omitted required quantities. *The input stops when* ITEM="END" is encountered.

After each read, the code first sets the values of the input parameters that can be set from stored values, if such is indicated, on the present read (Section I-B7).

After setting parameters from stored values, spacings are set from existing segments if called for, and the Cartesian coordinates for numbered points are taken from the array RPOINT(3, point number) if such is indicated (Section I-B9).

The code then checks for violation of dimension limits or bad input values. The plotter is then initialized if plotting is called for on the present read, and none has been done before, by calling the system routines COMPRS and CROSS, and then resetting the integer SETPLT from the default "NO" to "DONE" to indicate that the initialization has been done. The code next performs any needed conversion of direction cosines (values in the array COSINES greater than one in magnitude are taken to be angles in degrees and are replaced by the cosine thereof) and sets up storage files and/or cores for combination or treatment of multiple files or curves (Section I-B2).

Some operations, such as combination of files and/or cores and transformation or scaling of several surfaces, involve groups of files and/or cores. Consequently, COREOUT, COREIN, FILEOUT, and FILEIN are arrays dimensioned by the integers DCOR and DFIL, respectively. After each NAMELIST read, and before the action called for by the read is taken, the multiple files and/or cores are set up as follows.

The integer array CORES(4,DCOR), with 1 and 2 for the first subscript, contains the dimensions N1,N2 for each surface in core storage. The value 3 for this first subscript is used to store the core storage numbers, COREIN, for a group of stored surfaces to be combined or simultaneously scaled or transformed. Similarly the value 4 for the first subscript is used to store a group of core storage numbers, COREOUT, to receive the scaled or transformed results. The integer array FILES (4,DFIL) serves in the same manner for file storage.

The core numbers supplied in the array COREIN on the NAMELIST are placed successively in CORES with 3 as the first subscript and a counter as the second. Negative entries in COREIN imply all core numbers from the preceding entry to the magnitude of the negative entry. The total number of core locations in the group is placed in the integer NCORI. Core numbers given in the array COREOUT are placed successively in CORES with 4 as the first subscript in a similar fashion. If the first entry of COREOUT is equal to "SAME", then the same core numbers given in COREIN are taken for COREOUT. The total number of core locations in the group is placed in the integer NCORO. Analogous procedures are applied for a group of files.

Since the number of output cores(files) and input cores(files) must be the same, except in the "COMBINE" operation (Section I-B21), a check is made for inequality. Finally, CORIN and COROUT are set to the first entries in COREIN and COREOUT respectively (these are the core storage numbers that are transferred through COMMON/IO/ to subroutines GET and PUT), and NSTOI and NSTOO are set to the maximum number of files or cores for input and output, respectively.

C. RESPONSES TO NAMELIST/INPUT

The actions taken as a result of the read of the NAMELIST are as follows, depending on the value given for ITEM. These operations are explained in Part I. In the following discussion, reference to placing N points in an array A is made by referring to "the array A(N)". Upper-case letters thus indicate a set of elements. Similarly, the placing of three vector components for each of N points uses the terminology A(3,N). Specific elements are referred to using lower-case letters, i.e., A(n) means element n of A, and it should be clear from the context when a specific element is meant to be implied by a number. When no confusion should arise, the first element of an array A(2) is referred to simply as A, and similarly the first two elements of A(2,2) are referred to as A(2). Also, surface dimensions are often indicated in arrays such as CORD(3,N1,N2) for information only, and should not be taken to imply dimensions of the array (which are always DIM1,DIM2).

Several operations are used throughout the various actions resulting from the NAMELIST reads:

1. <u>Storing, printing, and plotting a surface</u> (Section I-B2)

This is accomplished by a call of subroutine

PUT(surface, dimension, dimension)

with the real surface array, CORD, as the first argument, and the integer dimensions of the surface transferred to the subroutine as the second and third arguments, typically N1,N2 for surfaces and N,1 for curves. Other parameters are transferred through COMMON/IO/. Core storage numbers are specified by the integer COREOUT, and file storage numbers by the integer FILEOUT, on the read. In some operations, multiple storage numbers can be given, and consequently these two parameters are actually integer arrays dimensioned DCOR and DIFL, respectively. The form of the file is controlled by FORM. Also REWOUT="NO" prevents rewinding a file before storage, and LABOUT="---" supplies an optional 8-character label to be printed. Printing and/or plotting is done if the array OUT(2) contains the values "PRINT" and/or "PLOT", in either order. For plotting, the limits of the plot can be given in the arrays RMIN(3) and RMAX(3), the viewpoint in the array VIEW(3), the size on the screen in the array SIZE(2), and a symbol specification in SYMBOL. Also if FRAME="NEW" is given, the plot will be on a new frame; otherwise it will be added to the previous plot to form a composite.

2. Retrieval of a surface (Section I-B3)

This is done by a call of subroutine

GET(surface, dimension, dimension)

with the real surface array, CORD or CORDI, as the first argument, and the integer dimensions of the surface transferred from the subroutine as the second and third arguments (typically N1,N2 or NI1,NI2). Other parameters are transferred through COMMON/IO/. Core numbers are specified by COREIN, and file numbers by FILEIN on the input. In some operations, multiple storage numbers can be given, and consequently these two parameters are actually integer arrays dimensioned DCOR and DFIL, respectively. The form of the file is controlled by FORM. Also, REWIN="NO" prevents rewinding a file before reading, and LABIN="---" supplies an optional 8-character label to be printed. Any entries in

VALUES will cause the surface to be read from this array in the NAME-LIST, instead of from core or file. (VALUES(1) is defaulted to "NONE" and is checked for a change on the read.)

3. <u>Relative distributions</u> (Section I-B6)

This is accomplished by a call of subroutine

RELDIST (type, spacings, interior point, dimension, distribution)

with the alphanumeric type of distribution as the first argument, a real array of two spacings as the second, the integer location of an interior point at which spacing is specified as the third, the integer number of points in the distribution as the fourth, and the returned real distribution array as the last. Of these arguments, the first (the distribution type) is supplied by the integer DISTYP(_), or one of the several similar parameters to which it is equivalenced. The spacings (a two entry array in the argument list) are given in the real array SPACE(2,_), or one of its equivalents. If an interior point spacing is specified, the arc length location thereof is the second entry in the array SPACE. The number of points in the distribution is set by POINTS(1), POINTSI(1), or some equivalenced parameter. The relative distribution is returned in the real array DISTRIB(DIM,_). In all of these arrays the absent entry is 1 or 2, this entry being 1 except when the distribution is being set for both directions on a surface.

4. Surface(Curve) Spline (Section I-B15)

Several operations can function on a curved surface as well as in space or on a plane. In this mode the surface must be constructed and splined before the operation in question is invoked. The spline is contained in the array SUR(3,DIM1,DIM2,0:3), where the last subscript refers to the points (0), the two tangents (1 and 2), and the cross-derivative (3) at each point on the spline. The spline is generated by SPLNSUR and is stored on file 8 by four successive calls of PUT with FILIN=-2 and CORIN=0 after rewinding. Whenever needed, the spline is retrieved by four analogous calls of GET. The input value of FILIN is preserved and restored after the spline is obtained.

5. ITEM="CURDIST" (Section I-E1)

This operation distributes N points on an existing curve having NI points, where N and NI are given as *POINTS* and *POINTSI*, *respectively*, with specified spacing at either or both ends, or at an interior point. The point distribution is determined by DISTYP and the array SPACE(2). The operation first calls GET to place the existing curve in the array CORDI(3,NI,1), and then calls ARCLNGT to calculate the arc length distribution on this curve in the array ARCLEN(NI). If RELATIV is equal to "NO", the values given in SPACE are taken as actual arc lengths and are therefore divided by the total arc length to produce relative spacings. Subroutine RELDIST is then called to set the N distribution factors in the array DISTRIB(N,1), and CURDIST is called to distribute the N points on the curve, placing the result in the array CORD(3,N,1).

The curve is splined in terms of chord length, and the point distribution is placed on the spline. If the curve has less than five points, the points are left on the chords since the spline is not possible. Finally, PUT is called to store, print, and/or plot the curve.

6. <u>ITEM="DISTRIB"</u> (Section I-B6)

This operation generates a set of N relative distribution factors, where N is given by POINTS and the distribution is determined by DISTYP and the array SPACE(2). If RELATIV is equal to "NO", the values given in the SPACE array are divided by the value given for TOTARC to obtain the relative spacings. Subroutine RELDIST is then called to set the distribution factors in the array DISTRIB(N,1).

7. <u>ITEM="CURVAT"</u> (Section I-E2)

This operation distributes N points on an existing curve with NI points, where N and NI are given as POINTS and POINTSI, respectively, concentrating the points where the curvature is greatest using a spline fit. The operation first calls GET to place the existing curve in the array CORDI(3,NI,1), and then calls SPARC to distribute the N points on the curve, placing the result in the array CORD(3,N,1). Finally, PUT is called to store, print, and/or plot the curve.

8. <u>ITEM="ARC"</u> (Section I-F2)

This operation calculates the arc length (actually chord length) distribution on a curve with N points, with N given as POINTS. Subroutine GET is called to place the curve in the array CORD(3,N,1), and ARCLNGT is then called to calculate the arc length distribution, placing the result in the array ARCLEN(N).

9. <u>ITEM="CURVEC"</u> (Section I-F3)

This operation calculates the unit tangent and principal normal, the curvature, and the arc length distribution on a curve with N points, N being given as POINTS. The operation calls GET to place the curve in the array CORD(3,N,1), and then calls CURVEC to calculate the various quantities on the curve, these being placed in the arrays TANGEN1(3,N,1), NORMAL(3,N,1), CURVAT(N), and ARCLEN(N), respectively.

10. <u>ITEM="SURVEC"</u> (Section I-F4)

This operation calculates the two unit tangents and normal on an N1xN2 surface, with N1 and N2 given as the two entries in the array POINTS(2). The operation calls GET to place the surface in the array CORD(3,N1,N2), and then calls SURVEC to calculate the three vectors, placing them in the arrays TANGEN1(3,N1,N2), TANGEN2(3,N1,N2), and NORMAL (3,N1,N2), respectively.

11. ITEM="GETEND" (SECTION I-E13)

This operation selects a point on an N-point curve, with N given as POINTS, for use as an end point of another curve to be constructed. The operation first calls GET to place the curve in the array CORD(3,N,1). The three Cartesian coordinates of the point on the curve given by POINT are then placed in the array REND1(3) if the value of END is given as "FIRST", and in REND2(3) otherwise.

12. ITEM="GENTAB"

This operation has two functions. If END is equal to "FIRST" or "SECOND" it selects a single point, and an associated unit tangent or unit normal, on a surface for later use as an end point of a cubic space curve (Section I-C2), or as a corner of a tensor-product surface (Section I-E8). Otherwise it selects a portion of a grid line, and the tangents or unit normals thereon, on the surface for later use as an end curve for a patch surface to be constructed of cubic curves (Section I-D3). In either case the surface is an N1xN2 surface, with N1 and N2 given as the two entries in the array POINTS(2), and the operation first calls GET to place the surface in the array CORDI(3,N1,N2).

In the first case, the selected point on the surface is indicated by the two indices given in the array POINT(2). Subroutine GENTAB is called to return the three Cartesian coordinates of the selected point in the array REND1(3) if END is equal to "FIRST", or in REND2(3) if END is "SECOND". Similarly, the three components of the slope vector at the point are returned in TEND1(3) or TEND2(3), the type of slope vector being determined by the alphanumeric value given for TABTYP. These vectors are then normalized to unit vectors.

In the other mode, the curve selected is defined by the coordinates of its end points given in the arrays START(2) and END(2) (with one entry the same in each), and the slope vectors are specified to be tangents or unit normals to the surface by the alphanumeric value given to TABTYP. In this second mode the normals are unit vectors, but the tangents reflect the spacing on the surface. The counter NTAB is incremented, and GENTAB is called to select the curve and generate the slope vectors, placing the curve in the array CURV(3,NI,NTAB) and the vectors in the array TABVEC(3,NI,NTAB), with NI equal to the difference between the non-equal entries in START and END, plus one, and being returned as POINTSI(1). Finally, this number of points is recorded in NTBPTS for later use.

13. <u>ITEM="EDGETAB"</u> (Section I-E10)

This operation functions as does the second mode for ITEM="GENTAB" above, except that the curve selected is an entire edge of the surface, the particular edge being indicated by the value given to EDGE.

14. <u>ITEM="CURTAB"</u>

This operation functions as does both modes of ITEM="GENTAB" above, selecting a point and associated slope vector for later use either with a cubic curve (Section I-C2) or a tensor product surface (Section I-E8), or with a patch curve (Section I-D3). In both cases the point selection is indicated by POINT. Here NI is 1, of course, since only a point is involved.

15. <u>ITEM="SETTAB"</u> (Section I-E14)

This operation directly sets a point and a unit slope vector as an end point for a cubic curve to be constructed, these being given in the arrays R(3) and T(3), respectively. The operation increments the counter NTAB, and places the point and slope vector in the arrays CURV(3,1,NTAB) and TABVEC(3,1,NTAB), respectively, setting NTBPTS to 1.

16. <u>ITEM="SETCOR"</u> (Section I-E8)

This operation sets a point and two tangents for use as one of the four corners of a tensor-product surface (Coon's patch). The three Cartesian coordinates of the point are received in the array R(3), and the components of the unit tangents in the arrays T1(3) and T2(3). If any of these three arrays is not included on the input, the corresponding values are taken from previously-set values in the arrays REND1(3) for R, or TEND1(3) for T1 and TEND2(3) for T2.

The counter NTAB is incremented, and the two unit tangents are multiplied by the spacings given in the array SPACE(2), and also by the total arc length TOTARC if these spacings are relative. The two indices of the corner given in the array POINT(2) are recorded in the array NCORPT(2,_) with NTAB as the second subscript.

If NTAB is equal to 4, i.e., if this is the last corner to be set, then the dimensions N1,N2 of the surface to be constructed are determined from the indices of the four corners in the array NCORPT(2,4) and are recorded in NCORPT(1,1) and NCORPT(2,2).

17. <u>ITEM="SCALE"</u> (Section I-E3)

This operation scales one or more surfaces(curves). The three scale factors, one for each Cartesian direction, are given in the array SCALE(3,1). For each surface in succession the operation first calls GET to place the surface in the array CORDI(3,N1,N2), with the storage location, FILIN or CORIN, being obtained from FILES(3,i) or CORES(3,i), as appropriate, for surface i, and the surface dimensions, N1 and N2, being taken from the arrays FILES(1,i) and FILES(2,i), or from CORES(1,i) and CORES(2,i) (Section II-C2). This surface is then scaled by calling SCAL, the scaled surface being placed in the array CORD(3,N1,N2). Subroutine PUT is then called to store, print, and/or plot the scaled surface. The storage location is obtained from FILES(4,i) or CORES(4,i), as appropriate. These steps are repeated to process all the surfaces indicated. Notations of the surfaces processed, and their storage locations and destinations, are printed.

18. <u>ITEM="TRANS"</u> (Section I-E4)

This operation functions exactly as does ITEM="SCALE" discussed above, except that TRANS, instead of SCALE, is called to translate and rotate, as well as scale, the surface(curve). The translation vector is given in the array ORIGIN(3), and the rotation is specified by either the nine direction cosines in the array COSINES(3,3) or the three Euler angles in the array EULER(3). If CHANGE is equal to "YES", an ad hoc change in the surface is made after the transformation. Code for this ad hoc change must be inserted directly in the code since no input provision is made.

19. <u>ITEM="CONICUR"</u> (Section I-C1)

This operation generates an N-point plane conic-section curve. The number of points is given as POINTS, the type of curve is identified by TYPE, and the relevant curve parameters are given as RADIUS, SEMIAX, LENGTH, WIDTH, ECCENT, and SYMTOT. The extent of the curve is specified by the two angles in the array ANGLE(2). The distribution of points on the curve is specified by DISTYP, INTER, and the array SPACE(2).

The operation first calls RELDIST to set the relative distribution factors in the array DISTRIB(N,1), and then calls CONICUR to set the points on the curve in the array CORD(3,N,1). Subroutine PUT is then called to store, print, and/or plot the curve.

20. <u>ITEM="SCURVE"</u> (Section I-C2)

This operation generates an N-point cubic space-curve between two points, with specified tangents at each end. The number of points is given as POINTS, the Cartesian coordinates of the end points in the arrays R1(3) and R2(3), and the corresponding unit tangents in the arrays T1(3) and T2(3). If any of these end point arrays is not included on the input, the corresponding values are taken from previously-assigned values in the arrays REND1(3), REND2(3), TEND1(3), or TEND2(3) (Section II-C12). The distribution of points on the curve is specified by DISTYP and the array SPACE(2).

This operation can function either in space (including on a flat surface) or on a previously constructed and splined curved surface. The latter mode is activated by the value "CURVED" being given for SURFACE. In this case the surface spline is obtained (Section II-C4), and the end points are reset to the closest points on the spline by calling CORPAR with R1 and R2 as arguments. This replaces the three Cartesian coordinates in R1 and R2 with two spline coordinates. Similarly, the three components of the tangents in T1 and T2 are converted to the two derivatives of the two spline coordinates in the two directions on the surface by calling TANPAR.

The operation then places the tangents in the array TABVEC(3,1,i) for i=1,2. Subroutine SCURVE is then called to generate the curve in the array CORD(3,N,1), and the curve is transferred to the array CORDI(3,N,1) by a call to GET. (If the operation is on a curved surface, the result produced by SCURVE is two surface parametric(spline) coordinates for each point on the curve. Therefore PARCOR is called to convert these to Cartesian coordinates of corresponding points on the spline.) Next, ARCLNGT is called to calculate the arc length distribution on the curve, returning the result in the array ARCLEN(N). If RELATIV is equal to "NO", the values in SPACE are taken to be arc lengths and are converted to relative spacings by division by the total arc length.

The total arc length, for multiplication by SPACE and the unit tangents to form r_ξ, is calculated as the straight line distance between the end points.

Subroutine RELDIST is then called to set the relative distribution factors in the array DISTRIB(N,1). Next POINTSI(1) is set to N, and CURDIST is called to place the points on the curve according to the specified distribution, returning the result in the array CORD(3,N,1). Subroutine PUT is then called to store, print, and/or plot the curve.

21. <u>ITEM="FLATSUR"</u> (Section I-D1)

This operation generates an N1xN2 plane surface bounded by a conic-section curve (closed by straight lines if necessary). The number of points on the circumferential curves is given as CURPTS, and that on the radial lines is given as RADPTS. The type of curve is identified by TYPE, and the curve parameters are given in RADIUS SEMIAX, LENGTH, WIDTH, and SYMTOT. The extent of the conic-section curve is set by two angles in the array ANGLE(2). The relative angular distribution of the points on the circumferential curves is specified by DISTANG and the array SPACCUR(2), and the relative radial distribution is specified by DISTRAD the array SPACRAD(2).

The operation calls RELDIST twice, once to set the relative distribution factors in the array DISTRIB(N1,1) for the points on the circumferential curves, and once to set the factors in DISTRIB(N2,2) for the points on the radial lines. Subroutine FLATSUR is then called to generate the surface in the array CORD(3,N1,N2), and PUT is called to store, print, and/or plot the surface.

22. <u>ITEM="CONISUR"</u> (Section I-D2)

This operation generates an N1xN2 conic-section surface, with the number of longitude lines given as LONPTS and the number of latitude lines by LATPTS. The type of surface is identified by TYPE, and the surface parameters are given as RADIUS, LENGTH, WIDTH, and ECCENT. The extent of the surface is set by two longitude angles and two latitude angles in the arrays LON(2) and LAT(2). The relative distribution of longitude lines is specified by DISTLON, INTLON, and the array SPACLON(2), and that for the latitude lines by DISTLAT, INTLAT, and the array SPACLAT(2).

The operation calls RELDIST twice, once to set the relative longitude distribution factors in the array DISTRIB(N1,1), and once to set the latitude factors in the array DISTRIB(N2,2). Subroutine CONISUR is then called to generate the surface in the array CORD(3,N1,N2), and PUT is called to store, print, and/or plot the surface.

23. <u>ITEM="SPLINE"</u> (Section I-E15)

This operation generates a cubic spline for an N1xN2 surface (curve). The surface is placed in the array CORD(3,N1,N2) by a call to GET, and then is splined by a call to SPLNSUR. The spline, in the array SUR(3,N1,N2,0:3), is stored on file 8.

24. <u>ITEM="INSEC"</u> (Section I-C5)

This operation generates a space-curve as the intersection of two surfaces. This intersection curve is composed of the intersections of one family of curves on an N1xN2 intersecting surface with an NI1xNI2 intersected surface. The intersected surface is placed in CORDI (3,NI1,NI2) by a call to GET using the storage location given as COREIN or FILEIN on the input. The intersecting surface is assumed to be in current position in CORD(3,N1,N2). The intersected surface in CORDI is splined by calling SPLNSUR, and INTSEC is called to calculate the intersections, returning the intersection curve in CORD(3,N1,1) with N1 points, i.e., the first dimension of the intersecting surface. Finally, PUT is called to store, print, and/or plot the curve.

25. <u>ITEM="EDGECUR"</u> (Section I-E7)

This operation designates a curve for later use as one of four edges of a surface to be constructed by transfinite interpolation. The counter NCUR is incremented, and MCUR is set to 1,2,3, or 4 depending on the particular edge for which the curve is intended, as indicated by

EDGE. The curve is placed in CURV(3,N1,_) by calling GET, and the number of points on the curve is recorded in NCORPT(1,_), with MCUR as the last subscript in each.

If NCUR is equal to 4, i.e., if this is the last curve needed, the dimensions N1,N2 of the surface to be constructed are set from NCORPT(1,3) and NCORPT(1,1).

26. ITEM="AXIS" (Section I-E6)

This operation prepares an N-point existing curve for use as the axis along which curves are stacked to produce a surface. The number of points on the axis curve is given as POINTS. The operation first calls GET to place the axis curve in the array AXIS(3,N), and then calls CURVEC to calculate the unit tangent, unit principal normal, and arc length, at each point thereon, these being returned in the arrays TANGEN1(3,N,1), NORMAL(3,N,1), and ARCLEN(N). The number of points on the axis is recorded in AXPTS.

27. ITEM="BOUNCUR" (Section I-E5)

This operation prepares an existing N-point curve for use as one of a set of two interpolation bounding curves for the creation of a surface by stacking, rotation or blending of curves. The number of points on the bounding curve is given as POINTS. The operation increments the counter NCUR and calls GET to place the bounding curve in the array CURV (3,N,NCUR). If NCUR is 1, the value "SAME" is placed in CURV(1,1,2) to default the second bounding curve to be identical to the first in case no second use of this operation is made. The number of points on the curve is recorded in NBNPTS.

28. <u>ITEM="ROTATE"</u> (Section I-D4)

This operation generates an N1xN2 surface by rotating N1-point space curves to N2 angles. The number of points on the rotated curves, N1, is obtained from NBNPTS, and the number of rotation angles, N2, is given by ANGPTS. The curve to be rotated is interpolated between two bounding curves that have been placed in the array CURV(3,N1,2) by previous invocations of ITEM="BOUNCUR". The angular limits of the rotation are given in the array ANGLE(2), and the direction cosines of the rotation axis and its principal normal are given in the arrays AXCOS(3) and NORCOS(3). The relative distribution of the rotation angles is specified by DISTANG and the array SPACANG(2). The relative distribution factors for the interpolation (between the two bounding curves) for the curve to be rotated is determined either by DISTCUR and the array SPACCUR(2), or from a specified distribution function.

The operation first converts the angular rotation limits to radians, defaulting the second angle if no value has been set. Subroutine RELDIST is called to set the relative distribution factors in the array DISTRIB(N2,1) for the rotation angles. The relative distribution factors for the interpolation (between the two bounding curves) for the curve to be rotated is determined in one of three ways: by DISTCUR and the array SPACCUR(2); from a specified distribution function; or by the arc length distribution on the axis.

If DISTCUR is equal to "CURVE", subroutine GET is called to place an existing NI-point characteristic size distribution function in the array SCALE(2,NI). In this case, the rotation angle distribution factors in the array DISTRIB(N2,1) are multiplied by the difference between the rotation limits (this difference being defaulted to 2π if no second limit is specified) to produce the N2 rotation angles which are placed in the array ARCLEN(N2). Subroutine FACDIST is then called to interpolate for the values of the normalized size distribution function (in SCALE) at these rotation angles (in ARCLEN), placing the relative distribution factors in the array DISTRIB(N2,2) for the interpolation between the two bounding curves. If, however, DISCUR is not equal to "CURVE", subroutine RELDIST is called to set these relative distribution factors in the array DISTRIB(N2,2) for the curve interpolation.

The surface is then generated in the array CORD(3,N1,N2) by calling ROTATE. If N1 is equal to 1, i.e., if a point is being rotated, SWITCH is called with REORDER(1)="SWITCH" to switch the curvilinear coordinates and put the resulting curve in CORD(3,N2,1). The counter NCUR for the bounding curves is reset to 0, and the surface is stored, printed, and/or plotted by calling PUT.

29. ITEM="STACK" (Section I-D5)

This operation generates an N1xN2 surface by stacking N1-point space curves at N2 positions along an axis. The number of points, N1, on the stacked curves is obtained from NBNPTS, and the number of positions, N2, on the axis is obtained from AXSPTS. The curve to be placed at each position is interpolated between two bounding curves that have been placed in the array CURV(3,N1,2) by previous invocations of ITEM= "BOUNCUR". The axis curve will have been placed in the array AXIS (3,N2) by a preceeding invocation of ITEM="AXIS". The direction cosines of the stacking axis and its principal normal in the axes system in which the bounding curves are defined are given in the arrays AXCOS(3) and NORCOS(3). The relative distribution factors for the interpolation (between the two bounding curves) for the curve to be stacked is determined in one of three ways: by DISTCUR and the array SPACCUR (2); from a specified distribution function; or by the arc length distribution on the axis.

If DISTCUR is equal to "CURVE", subroutine GET is called to place an existing NI-point characteristic size distribution function in the array SCALE(2,NI). In this case, FACDIST is then called to interpolate for the values of the normalized size distribution function (in SCALE) at the axis arc length positions (placed in the array ARCLEN(N2) by ITEM="AXIS"), placing the relative distribution factors in the array DISTRIB(N2,2) for the interpolation between the two bounding curves.

If DISTCUR is equal to "ARC", the relative distribution factors in DISTRIB(N2,2) are set equal to the relative arc length distribution of the points on the axis curve, this arc length distribution having been placed in the array ARCLEN(N2) by ITEM="AXIS". The relative distribution is obtained simply by dividing the arc length at each point on the axis by the total axis arc length.

If however, DISTCUR is not equal to "CURVE" or "ARC", subroutine RELDIST is called to set these relative distribution factors in the array DISTRIB(N2,2) for the curve interpolation.

The surface is then generated in the array CORD(3,N1,N2) by calling STACK; the counter NCUR for the boundary curves is reset to 0; and the surface is stored, printed, and/or plotted by calling PUT.

30. ITEM="BLEND" (Section I-D6)

This operation generates an N1xN2 surface by N2 interpolations between two N1-point space curves. The number of points, N1, on these two curves is obtained from NBNPTS, and the number of interpolated curves, N2, is given by CURVES. The two bounding curves between which the interpolation is done will have been placed in the array CURV (3,N1,2) by previous invocations of ITEM="BOUNCUR". The relative distribution factors for the interpolation between these two curves is determined either by DISTCUR and the array SPACCUR(2), or from a specified distribution function.

If DISTCUR is equal to "CURVE", subroutine GET is called to place an existing NI-point characteristic size distribution function in the array SCALE(2,NI). In this case, FACDIST is then called to interpolate for the values of the normalized size distribution function (in SCALE) at the integers from 1 to N2 (placed in the array ARCLEN(N2)), placing the relative distribution factors in the array DISTRIB(N2,2) for the interpolation between the two bounding curves. If, however, DISTCUR is not equal to "CURVE", subroutine RELDIST is called to set these relative distribution factors in the array DISTRIB(N2,2) for the curve interpolation.

This operation can function either in space or on a previously constructed and splined curved surface. The latter mode is activated by the value "CURVED" being given for SURFACE (Section I-B8). In this case the surface spline is obtained from file 8 (Section II-C4), and the two end points on the two bounding curves are reset to the closest points on the spline by calling CORPAR with CURV(3,N1,1) and CURV(,N1,2). This replaces the three Cartesian coordinates in CURV with two surface parametric(spline) coordinates.

The surface is then generated in the array CORD(3,N1,N2) by calling BLEND. If the operation is on a curved surface, the result produced by BLEND is two surface parametric(spline) coordinates for each point on the surface. Therefore in this case PARCOR is called to convert these to Cartesian coordinates of corresponding points on the spline. Finally, the counter NCUR for the bounding curves is reset to 0, and the surface is stored, printed and/or plotted by calling PUT.

31. ITEM="TRANSUR" (Section I-D7)

This operation generates an N1xN2 surface by transfinite interpolation from four curves forming the edges of the surface. The four edge curves will have been placed in the array CURV(3,_,4) by four previous invocations of the operation ITEM="EDGECUR". Here the second subscript is N1 or N2, depending on the edge, and these dimensions are obtained from NCORPT(1,3) and NCORPT(1,1) as set by the fourth "EDGECUR" usage.

This operation can function either in space or on a previously constructed and splined curved surface. The latter mode is activated by the value "CURVED" being given for SURFACE (Section I-B8). In this case the surface spline is obtained from file 8 (Section II-C4), and the points on the four edge curves are reset to the closest pints on the spline by calling CORPAR once for each edge curve. This replaces the three Cartesian coordinates in CURV with two surface parametric(spline) coordinates.

Next, TRANSUR is called to generate the surface in the array CORD(3,N1,N2). If the operation is on a curved surface, the result produced by TRANSUR is two surface parametric(spline) coordinates for each point on the surface. Therefore in this case PARCOR is called to convert these to Cartesian coordinates of corresponding points on the spline. The counter NCUR for the edge curves is reset to 0, and the surface is stored, printed and/or plotted by calling PUT.

32. <u>ITEM="TENSUR"</u> (Section I-D8)

This operation generates an N1xN2 surface by bi-cubic interpolation from the four corners in terms of set values for the corner points and two tangent vectors thereon. The four corner points and tangents will have been placed in the arrays CORD(3,_,_), TANGEN1(3,_,_) and TANGEN2(3,_,_) by four previous invocations of the operation ITEM=

"SETCOR". Here the last two subscripts in the arrays are (1,1), (N1, 1),(1,N2), and (N1,N2), these dimensions being obtained from NCORPT(1,1) and NCORPT(3,3) as set by the fourth use of "SETCOR".

This operation can function either in space or on a previously constructed and splined curved surface. The latter mode is activated by the value "CURVED" being given for SURFACE (Section I-B8). In this case the surface spline is obtained from file 8 (Section II-C4), and the points and tangents on the four corners are reset to the closest points on the spline by calling CORPAR once for each corner. The three components of the tangents are replaced by the derivatives of the two spline coordinates in the two directions on the surface.

Next, TENSUR is called to generate the surface in the array CORD(3,N1,N2). If the operation is on a curved surface, the result produced by TENSUR is two surface parametric(spline) coordinates for each point on the surface. Therefore in this case PARCOR is called to convert these to Cartesian coordinates of corresponding points on the spline. The counter NCUR for the corners is then reset to zero, and the surface is stored, printed and/or plotted by calling PUT.

33. <u>ITEM="PARCOR"</u> (Section I-D9)

This operation generates the Cartesian coordinates for points on an N1xN2 surface(curve) from the surface parametric(spline) coordinates of the points. The spline is obtained from file 8 (Section II-C4), and the surface parametric coordinates are placed in the array CORD(2,N1,N2) by calling GET. Then PARCOR is called to convert these coordinates to the Cartesian coordinates of the points, and PUT is called to store and print/or plot the surface.

34. <u>ITEM="PATCH"</u> (Section I-D3)

This operation generates an N1xN2 surface by connecting corresponding points on two tab curves with cubic curves. The number of points, N1, on the two tab curves is obtained from NTBPTS, and the number of points, N2, on the connecting cubic curves is given as POINTS. The two tab curves will have been placed in the array CURV(3,N2,2) by previous invocations of ITEM="GENTAB", "EDGETAB", "CURTAB", or "SETTAB". Also, a slope vector at each point on these tab curves will have been placed in the array TABVEC(3,N2,2).

This operation can function either in space or on a previously constructed and splined curved surface. The latter mode is activated by the value "CURVED" being given for SURFACE (Section I-B8). In this case the surface spline is obtained from file 8 (Section II-C4), and the points on the two patch curves are reset to the closest points on the spline by calling CORPAR with CURV(3,N1,1) and CURV(3,N1,2). This replaces the three Cartesian coordinates in CURV with two surface parametric(spline) coordinates. The surface is generated in the array CORD (3,N1,N2) by calling PATCH. Since PATCH functions with N1 and N2 reversed, SWITCH is called to reverse the dimensions of the surface obtained therefrom. If the operation is on a curved surface, the result produced by PATCH is two surface parametric(spline) coordinates for each point on the surface. Therefore in this case PARCOR is called to convert these to Cartesian coordinates of corresponding points on the spline. The counter NCUR for the tab curves is reset to 0, and the surface is stored, printed and/or plotted by calling PUT.

35. <u>ITEM="LINE"</u> (Section I-C4)

This operation generates an N-point straight line between two points in space. The number of points on the line is given as POINTS, and the end points are given as the arrays R1(3) and R2(3). If either of these is omitted from the input, the corresponding values are obtained from previously set values (Section I-B7) in the arrays REND1(3) and REND2(3). The point distribution on the line is determined by DISTYP and the array SPACE(2). If RELATIV is equal to "NO", the values given in SPACE are taken to be arc lengths and are converted to relative values by division by the length of the line. Subroutine RELDIST is then called to set the relative distribution factors in the array DISTRIB(N,1).

This operation can function either in space or on a previously constructed and splined curved surface. The latter mode is activated by the value "CURVED" being given for SURFACE (Section I-B8). In this case the surface spline is obtained from file 8 (Section II-C4), and the two end points are reset to the closest points on the spline by calling CORPAR with R1 and R2. This replaces the three Cartesian coordinates in R1 and R2 with two surface parametric(spline) coordinates. The line is generated in the array CORD(3,N,1) by calling LINE. If the operation is on a curved surface, the result produced by LINE is two surface parametric(spline) coordinates for each point on the surface. Therefore in this case PARCOR is called to convert these to Cartesian coordinates of corresponding points on the spline. The line is then stored, printed, and/or plotted by a call to PUT.

36. ITEM="CURRENT" (Section I-E16)

This operation places an N1xN2 point surface(curve) into the array CORD(3,N1,N2) by calling GET. Subroutine PUT is called to store, print, and/or plot the surface.

37. ITEM="SWITCH" (Section I-E18)

This operation reverses the order of the points and/or switches the two indices for an N1xN2 surface(curve). The actions intended are specified by one, two, or three entries in the array REORDER(3). Subroutine GET is called to place the surface in the array CORD(3,N1,N2), after which SWITCH is called to accomplish the actions. The revised surface in the array CORD(3,N1,N2) is stored, printed, and/or plotted by a call to PUT.

38. ITEM="OUTPUT" (Section I-E17,B-5)

This operation stores, prints, and/or plots an N1xN2 point surface(curve). Subroutine GET is called to place the surface in the array CORD(3,N1,N2), and PUT is called to store, print, and/or plot the surface.

39. <u>ITEM="INSERT"</u> (Section I-E19)

This operation inserts an NI1xNI2 point surface(curve) into an N1xN2 point surface(curve). The position on the second surface at which the insertion is made is specified by the two entries in the array START(2).

The operation calls GET to place the first surface into the array CORDI(3,NI1,NI2). (The second surface will already be in current position, i.e., in the array CORD(3,N1,N2)). The insertion is then made by calling INSERT, the composite surface being returned in the array CORD (3,N1,N2) where N1 and N2 will have been increased if necessary to accomodate the insertion. The composite surface is stored, printed, and/or plotted by calling PUT.

40. <u>ITEM="EXTRACT"</u> (Section I-E20)

This operation extracts an N1xN2 point surface(curve) from an NI1xNI2 point surface(curve). The dimensions of the former are given as the two entries in the array POINTS(2). The position on the surface from which the extraction is made is specified by the two entries in the array START(2).

The operation calls GET to place the second surface into the array CORDI(3,NI1,NI2), and then calls EXTRACT to make the extraction into the array CORD(3,N1,N2). If N1 is equal to unity, SWITCH is called with REORDER(1) = "SWITCH" to switch the indices so that the second, instead of the first, is unity. Subroutine PUT is called to store, print, and/or plot the extracted surface.

41. ITEM="COMBINE" (Section I-E21)

This operation combines several files and/or cores onto a single file or core. The file and/or core numbers to be combined are given on the input in the arrays FILEIN(_) and COREIN(_). The output file or core is given in FILOUT or COROUT.

If the combination is to be put on a file, each input file or core is written directly onto the output file. If the combination is to be to a core storage, each input file or core is first written onto file 9, and this file is then transferred to the output core. This is accomplished by setting FILOUT to -1 and COROUT to 0, having first preserved the input value of COROUT and any input values in OUT for printing and/or plotting. Since an output file here is not to be rewound after each file or core is added, REWOUT is set to "NO". The point counter NN is set to 0. Each file, and then each core, in succession is placed in the array CORD(3,N1,N2) by a call to GET, the point counter NN is incremented by N1*N2, PUT is called to add the file or core to the output file or file 9, and a printed notation of the addition is made.

For output to a file, the format is controlled by TRIAD and FORM (Section I-B2). If CONTENT="YES", a table of contents is written at the beginning of the file, containing the COREIN number and the dimensions N1,N2 of each surface(curve) on the file. The format of this table of contents is also controlled by FORM, with COREIN,N1,N2 on a single line.

If the output is to core storage, the value of the output core number, COROUT, and the printing and/or plotting indicators in OUT are restored and file 9 is rewound. The combined input files and/or cores are then placed in CORD(3,NN,1) by a call to GET with FILIN=-1 to read from file 9, after which PUT is called with FILOUT set to 0 to place the combined files and/or cores into a single core storage. Finally, FILES(1,FILOUT), and/or CORES(1,COROUT), is set to the total number of points on all the files, and/or cores, and FILES(2,FILOUT) and/or CORES(2,COROUT) is set to unity.

42. <u>ITEM="COPY"</u> (Section I-E22)

This operation copies one or more files and/or cores onto other files and/or cores. The files and/or cores to be copied are given on input in the arrays FILEIN(_) and COREOUT(_), while those for the copies are given in the arrays FILEOUT(_) and COREOUT(_). Each file, and then each core, in succession is placed in the array CORD(3,N1,N2) by calling GET and then is copied to the next output file or core in succession by a call to PUT. A printed notation of the copying is made.

43. <u>ITEM="POINT"</u> (Section I-E25)

This operation sets the three Cartesian coordinates of a numbered point. The point number is given by POINT, and the coordinates are given either explicitly as R(3) or must have been set by a previous invocation of the operation ITEM="GETEND" (Section II-C11). In the latter case the coordinates are transferred from REND1. In either case the coordinates are placed in the array RPOINT(3, point number).

44. ITEM="CORPAR" (Section I-E26)

This operation determines the values of the surface parametric(spline) coordinates for the points on an N1xN2 surface(curve) generated on a curved surface. The spline is obtained from file 8 (Section II-C4), and the Cartesian coordinates of the surface(curve) are placed in the array CORD(3,N1,N2) by calling GET. Then CORPAR is called to convert these coordinates to the parametric coordinates of the points, and PUT is called to store and print/or plot the parametric values.

45. ITEM="SETVAL" (Section I-E23)

This operation stores a real value for later use. The value can be given directly, or can be calculated as a sum, difference, or product of other values as indicated by MATH. The terms in the calculation are in the array TERMS, and the value is stored in the array RVALUE(VALOUT). Negative entries in TERMS indicate previously stored values, with the magnitude giving the location in RVALUE.

46. ITEM="SETNUM" (Section I-C1 of Vol. III)

This operation stores an integer value for later use. The value can be given directly, or can be calculated as a sum, difference, or product, or as variations of these as discussed in the indicated section of Vol. III. The type of calculation is indicated by MATH, and the terms of the calculation are in ITERMS. The value is stored in IVALUE(VALOUT). Negative entries in ITERMS indicate previously stored values, with the magnitude giving the location in IVALUE.

47. <u>ITEM="SURDIST"</u> (Section I-E27)

This operation distributes N1xN2 points on an existing surface having NI1xNI2 points, where N1,N2 are given as POINTS and NI1,NI2 are given as POINTSI. The point distribution is determined by the arrays DISTYP(2) and SPACE(2,2).

The operation extracts each of the four edges from the surface in succession and uses the "CURDIST" operation to distribute points on the edges. The surface is then splined and CORPAR is called on each edge to convert to parametric coordinates. TRANSUR is then called to create a lattice of parametric coordinates, and finally PARCOR is called to convert the lattice back to Cartesian coordinates on the surface. Finally, PUT is called to store, print, and/or plot the surface.

SUBROUTINES LIST

SUBROUTINE . . . . . Page

ACOSG 159
AITKEN 159
ARCLNG 160
BLEND 190
CONICUR 168
CONISUR 173
CORPAR 218
COSG 159
CURDIST 157
CURSPL 226
CURTAB 209
CURVEC 160
DET3 172
DET4 172
EDGETAB 194
EXTRACT 207
FACDIST 211
FLATSUR 172
GENTAB 164
GET 202

From core 203
From file 203
From namelist 205
From the current array 205

INSERT 206
INTERP1, INTERP2, and INTERP3 197
INTSEC 178
LINE 197
PARCOR 224
PUT 198

Core Storage 198
File Storage 200
Print 201
Plot 201

PATCH 191
RELDIST 157
ROTATE 181
SCAL 167
SCURVE 208
SING 159
SPARC 213
SPLINE 212
SPLNSUR 225
STACK 186

SURSPL 225
SURVEC 162
SWITCH 205
TANG 159
TANPAR 222
TENSUR 216
TRANS. 167
TRANSUR. 214

D. SUBROUTINES

1. CURDIST

This subroutine distributes points on a given curve at a specified distribution of relative arc lengths using linear interpolation. The given curve, with NI points, and the arc length distribution on this curve are in the arrays RI(3,NI) and AI(NI), respectively. The relative arc length distribution (monotonic variation from 0 to 1) for the N points on the output curve is received in the array AO(N). The routine first multiplies AO by the total arc length on the given curve, and then places N points on that curve by linear interpolation for each value in AO among those in AI.

The input point distribution is splined in terms of chord-length, with the spline type specified by TYPE. (The default is a quadratic spline.) The designated number of points then are distributed on the spline curve according to the relative chord-length distribution set by subroutine RELDIST.

2. RELDIST

This subroutine sets a relative distribution of N (an integer argument) values, monotonic on the range 0-1. The distribution may be linear or with specified intervals on one or both ends, or at an interior point, according to a hyperbolic tangent or hyperbolic sine function (Appendix A). These hyperbolic functions produce a distribution which reduces the truncation error that arises from the rate-of-change of the point spacing (Ref. 4-5).

The type of distribution is received as the alphanumeric argument DISTYP:

"LINEAR":	linear distribution (no specified intervals).
"BOTH":	specified intervals at both ends using the hyperbolic tangent distribution.
"TANH":	specified interval at the first end using the hyperbolic tangent distribution. (This gives the smoothest distribution over the entire range.)
"SINH":	specified interval at the first end using the hyperbolic sine distribution. (This gives the most uniform distribution near the specified spacing.)
"INTERIOR":	specified interval at an interior point using the hyper bolic sine. Here both the distribution fraction and the point number at which this interval is to occur are specified also.

The specified intervals at point 1 and point N (if applicable) are received in the real argument SPACE as SPACE(1) and SPACE(2), respectively. The specified interval at an interior point (if applicable) is received in SPACE(1), with the distribution fraction being received in SPACE(2). The distribution is returned in the real argument array FRAC(N) in all cases.

The linear distribution is given by

$$\mathrm{FRAC}(i) = \frac{i-1}{N-1} \qquad i = 1,2,\text{--},N$$

and the hyperbolic function distributions are generated from the equations given in Appendix A. The code notation follows closely that of this appendix.

3. SING, COSG, TANG, ACOSG

These functions return the hyperbolic sine, cosine, tangent, and inverse cosine if FLAG (transferred through COMMON/HYPER/) is less than unity for the first three, and if the magnitude of the argument is greater than unity for the last. Otherwise the circular functions are returned. The angle is received as the real argument ANG in radians for the first three and as the real value A for the last.

4. AITKEN

This subroutine receives the current and preceding values of an iterate and the residual in the solution of a nonlinear equation and returns a new iterate for Newton-Raphson iteration. The current iterate and residual are received as the real arguments X and F, with the preceeding values in the real arguments XO and FO. The new iterate, returned in X, is calculated from

$$X = X - AC \left(\frac{X - XO}{F - FO}\right) F$$

where a parameter (range 0-1) is received as the real argument AC. (At the first call, X is simply returned as 1.01X.) The quotient in parentheses is bounded by unity and is also set to unity when the residual falls below a set minimum (FMIN=10^{-8}). Each iterate will be printed if the integer argument IPRT is nonzero.

5. ARCLNGT

This subroutine receives a space curve of N (integer argument) points in the real argument array R(3,N), and returns the arc length at each point in the real argument array ARC(N). This arc length is calculated as the chord length:

$$\text{ARC}(i) = \text{ARC}(i-1) + |\underset{\sim}{r}(i) - \underset{\sim}{r}(i-1)| \qquad i = 2,3,\text{--},N$$

with ARC(1) = 0.

6. CURVEC

This subroutine receives an N-point (integer argument) space curve in the real argument array R(3,N), and returns the unit tangent, unit principal normal, curvature, and arc length at each point in the real argument arrays TNG(3,N), NOR(3,N), CUR(N), and ARC(N). (If the alphanumeric argument LUNIT is equal to "NO", the tangent is not a unit tangent, but will reflect the spacing on the curve).

The arc length, s, is calculated as the chord length:

$$s(i) = s(i-1) + |\underset{\sim}{r}(i) - \underset{\sim}{r}(i-1)| \qquad i = 2,3,\text{--},N$$

with s(1) = 0.

The unit tangent, $\underset{\sim}{T}$, is given by the derivative of $\underset{\sim}{r}$ with respect to arc length:

$$\underset{\sim}{T} = \frac{d\underset{\sim}{r}}{ds}$$

which is represented at the interior points using second-order central differences:

$$T(i) = \frac{\underset{\sim}{r}(i+1) - \underset{\sim}{r}(i-1)}{s(i+1) - s(i-1)} \qquad i = 2,3,--,N-1$$

and at the ends using first-order one-sided differences:

$$\underset{\sim}{T}(1) = \frac{\underset{\sim}{r}(2) - \underset{\sim}{r}(1)}{s(2) - s(1)}$$

$$\underset{\sim}{T}(1) = \frac{\underset{\sim}{r}(2) - \underset{\sim}{r}(1)}{s(2) - s(1)}$$

The principal normal, $\underset{\sim}{N}$, is given by the derivative of the tangent with respect to arc length:

$$\underset{\sim}{N} = \frac{d\underset{\sim}{T}}{ds}$$

which is represented in difference form analogously to that given above for the tangent. The curvature, κ, is calculated as the magnitude of the principal normal:

$$\kappa = |\underset{\sim}{N}|$$

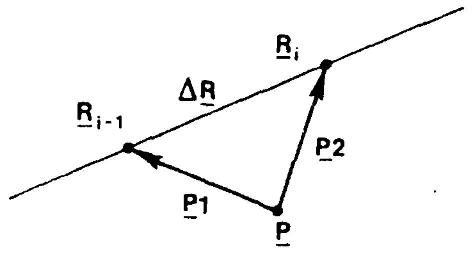
The principal normal is then made a unit normal by division by this curvature.

At points where the curvature is less than 10^{-8}, the unit principal normal is taken as directed toward a set point off the curve. This point is received in the real argument array P(3) if the curve is a straight line; otherwise it is located by adding $\underset{\sim}{N}$ to $\underset{\sim}{r}$ at the last point having curvature greater than 10^{-8}. If the first point has curvature less than 10^{8}, the normal there is set to that at the second point.

This is done as follows: With a point off the line set in the array P(3), the two vectors P1(3) and P2(3) are formed as

$$\underline{P1} = \underline{R}_{i-1} - \underline{P}$$

$$\underline{P2} = \underline{R}_i - \underline{P}$$



The normal then is taken as

$$\begin{aligned} \underline{N} &= \Delta\underline{R} \times (\underline{P2} \times \underline{P1}) \\ &= (\underline{P2} - \underline{P1}) \times (\underline{P2} \times \underline{P1}) \\ &= [(\underline{P1} \cdot \underline{P2}) - |\underline{P2}|^2]\underline{P1} + [(\underline{P1} \cdot \underline{P2} - |\underline{P1}|^2]\underline{P2} \end{aligned}$$

If the straight line follows a curved portion, the point $\underline{P}$ is taken as $\underline{R}_{i-1} + \underline{N}_{i-1}$. Otherwise it is received in the array P(3).

Finally, if LUNIT = "NO", the unit tangents are multiplied by the arc length spacing.

7. SURVEC

This subroutine receives an N1xN2 (integer arguments) surface in the real argument array R(3,N1,N2) and returns the unit tangents to the grid lines on the surface, and the unit normal to the surface, in the real argument arrays TAN1(3,N1,N2), TAN2(3,N1,N2), and NOR(3,N1,N2).

(If the alphanumeric argument LUNIT is set to "NO", the tangents will not be normalized to unity, but will reflect the spacing on the surface).

The tangents are given by the increment in $\underset{\sim}{r}$ along the respective curvilinear coordinate lines:

$$\underset{\sim}{T}^1 = \Delta_1\underset{\sim}{r}, \qquad \underset{\sim}{T}^2 = \Delta_2\underset{\sim}{r}$$

These are calculated by central differences at interior points:

$$\underset{\sim}{T}^1(i,j) = \frac{\underset{\sim}{r}(i+1,j) - \underset{\sim}{r}(i-1,j)}{2} \qquad \begin{array}{l} i = 2,3,\text{--},N1-1 \\ j = 1,2,\text{--},N2 \end{array}$$

$$\underset{\sim}{T}^2(i,j) = \frac{\underset{\sim}{r}(i,j+1) - \underset{\sim}{r}(i,j-1)}{2} \qquad \begin{array}{l} i = 2,3,\text{--},N1 \\ j = 2,3,\text{--},N2-1 \end{array}$$

Here i and j refer to the second and third indices, respectively, in the array R(3,N1,N2). The edge values are calculated by one-sided differences:

$$\left.\begin{array}{l} \underset{\sim}{T}^1(1,j) = \underset{\sim}{r}(2,j) - \underset{\sim}{r}(1,j) \\ \underset{\sim}{T}^1(N1,j) = \underset{\sim}{r}(N1,j) - \underset{\sim}{r}(N1-1,j) \end{array}\right. \qquad j=1,2,\text{--},N2$$

and

$$\left.\begin{array}{l} \underset{\sim}{T}^2(i,1) = \underset{\sim}{r}(i,2) - \underset{\sim}{r}(i,1) \\ \underset{\sim}{T}^2(i,N2) = \underset{\sim}{r}(i,N2) - \underset{\sim}{r}(i,N2-1) \end{array}\right. \qquad i=1,2,\text{--},N1$$

Zero values of $\underset{\sim}{T}^1$ on j=1 and j=N2 are replaced by values on j=2 and j=N2-1, respectively. (This situation can occur when one edge of the surface degenerates to a point in physical space, e.g., a polar axis.) An analogous procedure applies for $\underset{\sim}{T}^2$ on i=1 and N1.

If LUNIT is not equal to "NO", these tangents are then converted to unit tangents by division by the magnitudes. The unit normal, $\underset{\sim}{N}$, to the surface is calculated from

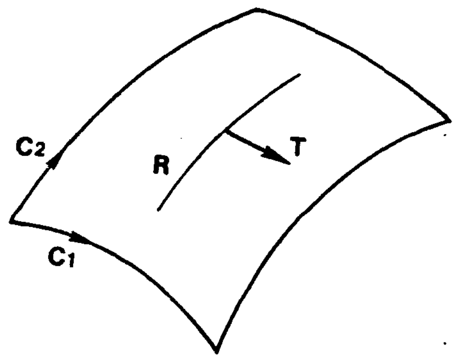
$$\underset{\sim}{N} = \frac{\underset{\sim}{T}^1 \times \underset{\sim}{T}^2}{|\underset{\sim}{T}^1 \times \underset{\sim}{T}^2|}$$

8. GENTAB

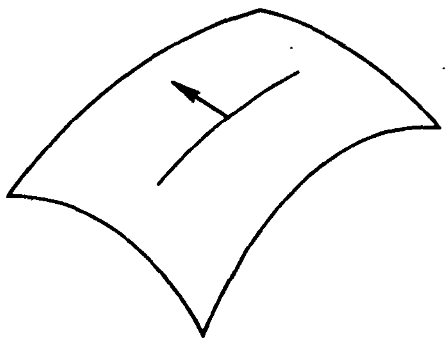
This subroutine receives an N1xN2 surface in the array RI(3,N1,N2), and returns an N-point portion of one of the curves on the surface in the real argument array R(3,N), and a set of slope vectors in the real argument array T(3,N), to be used to join another surface intersecting the given surface on the curve R with slope T. The surface dimensions, N1 and N2, are equivalenced with POINTS, and RI is equivalenced with CORDI, which are received through COMMON/PAR/ and COMMON/RAY/, respectively.

The vectors in T may be tangents to the curves crossing the curve R on the surface or normals to the surface, as specified by the following values of TABTYP(received through COMMON/PAR/):

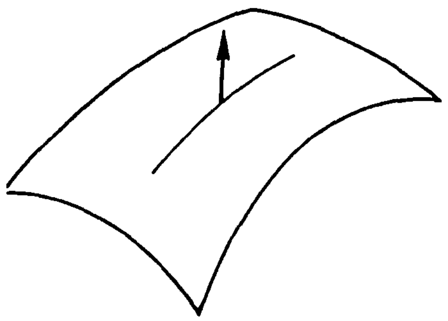
"TANPOS": positive tangent to crossing curve:



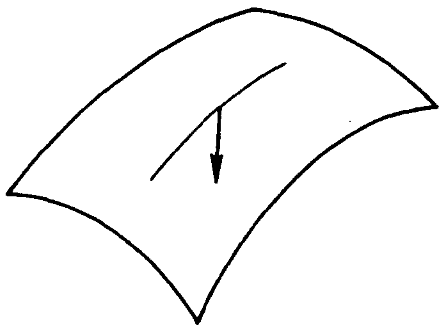
"TANNEG": negative tangent:



"NORPOS": positive unit normal to the surface:



"NORNEG": negative unit normal:



The indices of the ends of the curve R on the surface are in I1(2) and I2(2), equivalened with START and END, repectively, which are received through COMMON/PAR/. Either the first or second values in I1 and I2 must be equal, of course.

First the tangents, TAN1 and TAN2, to the two sets of curves on the surface, and the normal, NOR, to the surface, are determined by calling subroutine SURVEC for the surface in RI. These tangents are not unit vectors but rather reflect the spacing on the surface. In the following discussion, i and j will be used to represent the second and third indices in RI, i.e., the two coordinates on the surface.

If the curve R on the surface is to be part of a curve on which j is constant, then the number of points on this curve, N, is determined from

$$N = |I2(1) - I1(1)| + 1$$

The points on the curve segment, R, are set from RI with j=I1(2)=I2(2) for i from I1(1) to I2(1). If the positive (negative) tangent is indicated, T is set from (-) TAN2 in like manner, else for the positive (negative) normal, T is set from (-)NOR. An analogous procedure is followed if the curve segment is on a curve of constant i.

This subroutine can also treat a plane curve, rather than a surface, with the plane curve being received in RI(3,N1,1). In this case, a surface is first created by setting values in RI(3,i,2) and RI(3,i,3) equal to those in RI(3,i,1) plus 1 and 2, respectively, with the other components in RI on j=2 and 3 equal to those for j=1.

9. SCAL

This subroutine receives an N1xN2 surface in the array RI(3,N1,N2), and a set of three scale factors (one for each Cartesian direction) in the array SCALE(3), and outputs a scaled surface in the array R(3,N1,N2). The scale factors are received through COMMON/RAY/. The scaling is done by multiplying each component in RI by the corresponding scale factor. The surface in RI and its dimensions, N1 and N2, are received through equivalence with CORDI and POINTS in COMMON/RAY/ and COMMON/PAR/, and the scaled surface in R is returned through equivalence with CORD in COMMON/RAY/.

10. TRANS

This subroutine receives an N1xN2 surface in the array RI(3,N1,N2), and outputs a scaled and transformed surface in the array R(3,N1,N2). The surface in RI and its dimensions, N1 and N2, are received through equivalence with CORDI and POINTS in COMMON/RAY/ and COMMON/PAR/, and the scaled surface in R is returned through equivalence with CORD in COMMON/RAY/.

The transformation positions the origin of the axes relative to which the input surface is defined at the location $\mathbf{r}_0$ in the output system, $\mathbf{r}_0$ being received in the array ORIGIN(3). The input axes system is rotated according to either direction cosines or Euler angles, nine values being received in the array PARAM(3,3) for the former and three values is in PARAM(i,1), with i=1,2,3, for the latter. (The Euler angles are recognized when all of the last six entries in PARAM are zero.) The rotation matrix A(3,3) is first filled as appropriate with the direction cosines, either directly from PARAM or from the relations in terms of the Euler angles. The transformation then is performed according to

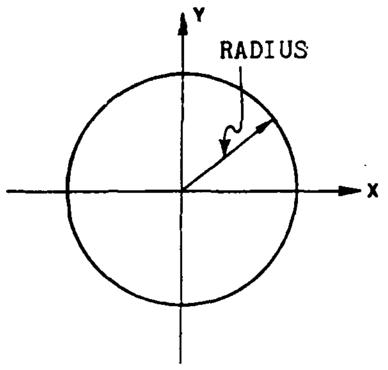
R = ORIGIN + A * (RI * SCALE)

so that scaling is done first, followed by rotation and then translation.

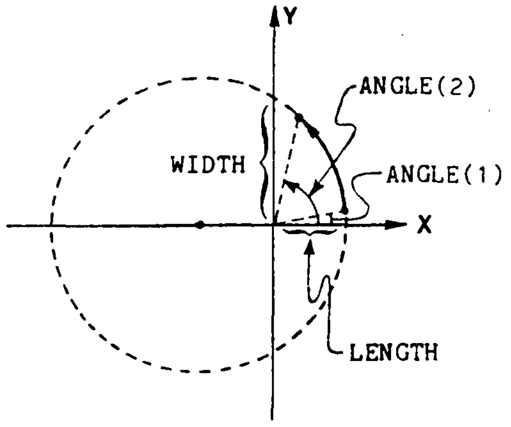
11. CONICUR

This subroutine generates a conic section curve in the x-y plane (with z=0). The type of curve is specified by the alphanumeric integer TYPE as follows, with the real parameters shown in each case:

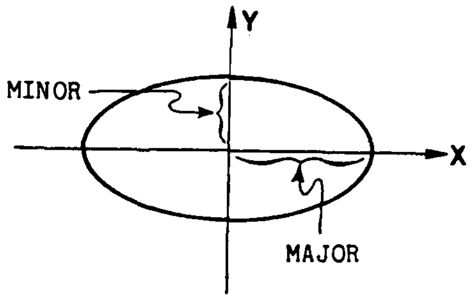
TYPE = "CIRCLE" - circle



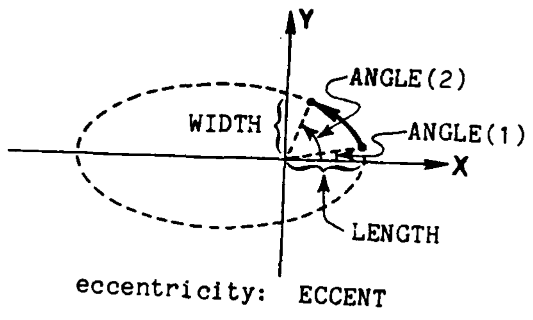
TYPE = "CIRARC" - circular arc



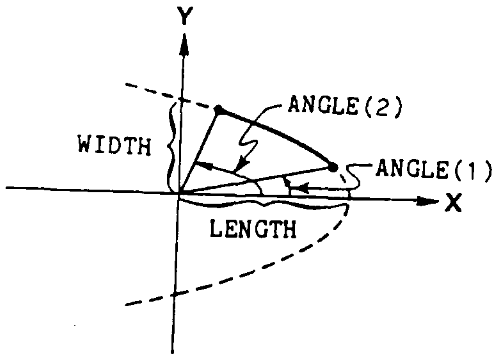
TYPE = "ELLIPSE" - ellipse



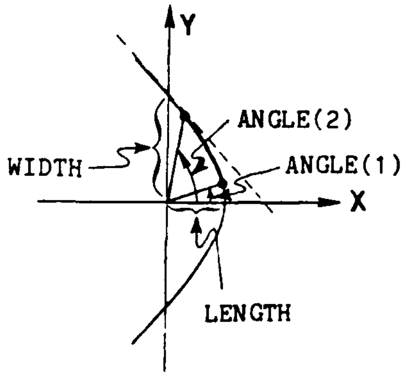
TYPE = "ELLIARC" - elliptical arc



TYPE = "PARABOLA" - parabola



TYPE = "HYPERBOL" - hyperbola



The closed curves are centered at the origin. The ellipse has its axes on the x and y axes, and the axis of the parabola and hyperbola are on the x-axis, with the vertex on the positive x-axis.

The conic section curve parameters are received through COMMON/PAR/, either directly or through equivalence. The number of points on the curve is specified by the integer N (equivalenced with POINTS in COMMON/PAR/), and these points run from the position determined by ANGLE1 to that of ANGLE2, these being equivalenced with ANGLE in COMMON/PAR/. These angles are measured counter-clockwise from the positive x-axis as shown, and either angle may be the larger. If no angles are given, the points on the two closed curves run counter-clockwise from the intersection on the positive x-axis, and the points on the open curves run from the intersection on the negative y-axis to that on the positive y-axis.

A relative angular distribution can be specified in the real array ANGDIS(N), with monotonic values from 0 to 1, received through equivalence with DISTRIB(i,1), for i=1,2,...,N, in COMMON/RAY/. The points will be located at the angle given by

ANG = ANGLE1 + ANGDIS(i) * (ANGLE2 - ANGLE1)

for i = 1,2,--,N. The angles will be equally spaced if ANGDIS is not specified. The routine converts angles to radians. If no limiting angles are specified, ANG1 is set to 0 and ANG2 is set to 2π for the closed curves, or to $-\pi/2$ and $\pi/2$ for the open curves.

At each point, the radius is calculated as a function of the angle from the equations given in Appendix B, and is placed in the real array RAD(N). The points on the curve are then placed in the real array

R(3,N), with the Cartesian component index as the first subscript, for return through equivalance with CORD in COMMON/RAY/. Thus, for i=1,2....,N,

```
R(1,i) = RAD(i) * COS(ANG)
R(2,i) = RAD(i) * SIN(ANG)
R(3,i) = 0
```

with ANG calculated for each i as noted above.

12. DET3

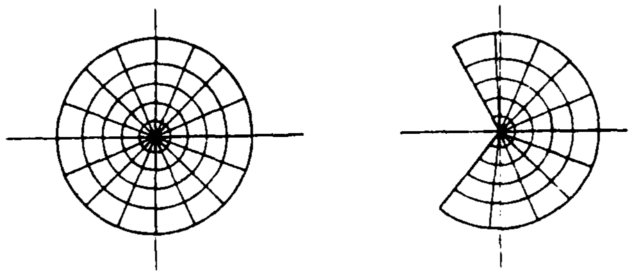
This function evaluates a 3x3 determinant from Kramer's rule.

13. DET4

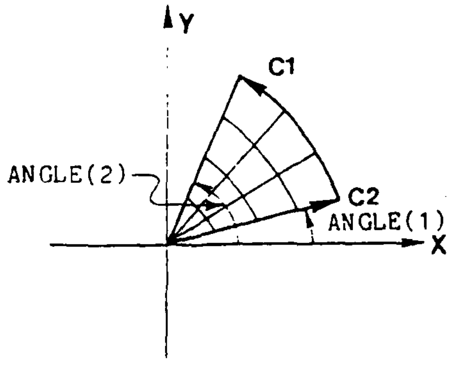
This function evaluates a 4x4 determinant by expansion in co-factors, using DET3 to evaluate the 3x3 determinants.

14. FLATSUR

This subroutine generates a surface in the x-y plane (with z=0) bounded by a conic-section curve. The curve is generated exactly as in subroutine CONICUR, and the open curves are closed by straight lines connecting the end points to the origin. The grid is formed by connecting points on radial lines from the origin:



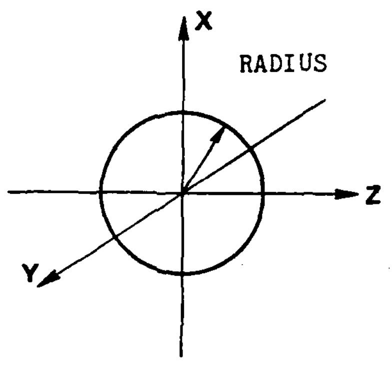
The number of points on the curve is specified by NANG, and that on the radial lines by NRAD, these being equivalenced with POINTS in COMMON/PAR/. Relative distributions on both the curve and the radial lines can be specified. These distributions are in the arrays ANGDIS(NANG) and RADDIS(NRAD), received through equivalance with DISTRIB(i,1) and DISTRIB(i,2) in COMMON/RAY/, with i=1,2...,NANG and j=1,2...,NRAD. The points are placed on the radial lines by multiplication of the r(θ) from Appendix B by RADDIS(j) for j=1,2,--,NRAD. The surface points are placed in the array R(3,NANG,NRAD) for return through equivalence with CORD in COMMON/RAY/. The first index of the grid on the surface varies on the curves, while the second varies on the radial lines:



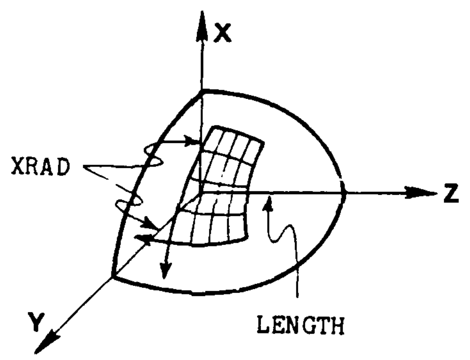
15. CONISUR

This subroutine generates a conic-section surface. The type of surface is specified by the alphanumeric integer TYPE as follows, with the real conic-section parameters shown in each case:

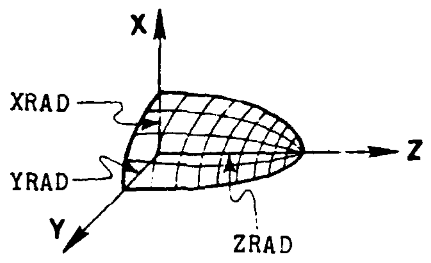
TYPE = "SPHERE" - sphere



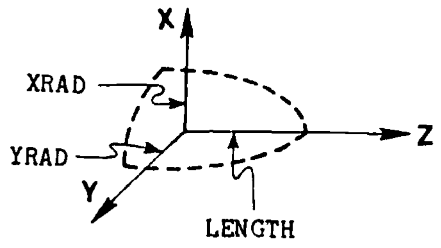
TYPE = "SPHSEG" - spherical segment



TYPE = "ELIPSOID" - ellipsoid

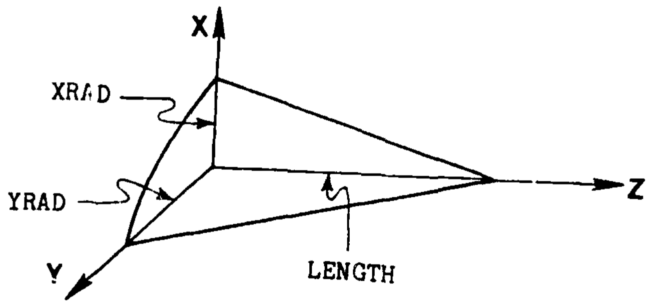


TYPE = "ELLISEG" - ellipsoidal segment

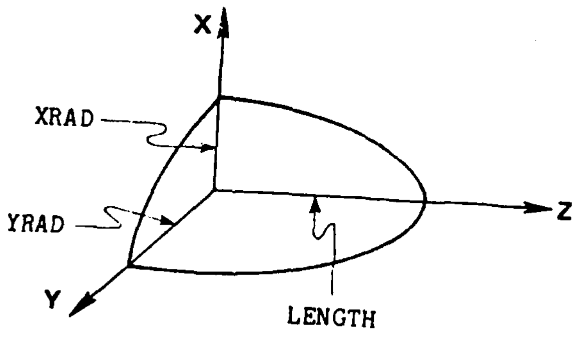


ECCENT: eccentricity in x-z plane

TYPE = "ELLICONE" - elliptic cone

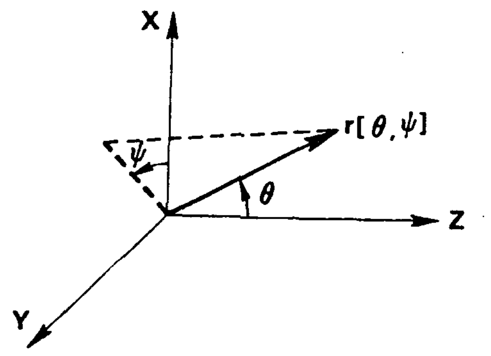


TYPE = "ELLIPAR" - elliptic paraboloid



The closed surfaces are centered on the origin. The ellipsoid has its axes on the three coordinate axes, and the cone and paraboloid have the axis on the z-axis, with the vertex on the positive z-axis. The conic section surface parameters are received through COMMON/PAR/, either directly or through equivalence.

The grid on the surface is formed by lines at constant θ and ψ, analogous to latitude and longitude, respectively, with the z-axis as the polar axis:



The number of latitude lines, i.e., values of θ, is specified by the integer NLAT, and that of the longitude lines by NLON. The points on the latitude lines run from the longitude angle specified by the real variable LON1 to longitude angle LON2. Similarly the points on the longitude lines run from the latitude angle LAT1 to LAT2. In each case, either of the two angles may be the larger. If no angles are specified, θ runs from 0 to π, for the closed curves and from 0 to π/2 for the open curves, and ψ runs from 0 to 2π, placing the points accordingly. These parameters are equivalanced with POINTS, LON, and LAT in COMMON/PAR/.

A relative angular distribution can be specified for each angle by the real array LATDIS(NLAT) for latitude, i.e., θ, and by LONDIS(NLON) for longitude, ψ. Each of these arrays contains monotonic values from 0 to 1, received through equivalence with DISTRIB(j,2) and DISTRIB(j,1), in COMMON/RAY/, with j=1,2...,NLAT and i=1,2,...,NLON. Latitude lines are located at values of θ given by

θ: LAP = LAT1 + LATDIS (j) * (LAT2 - LAT1)

for j=1,2,--,NLAT. Similarly, longitude lines are located at ψ values given by

ψ: LOP = LON1 + LONDIS (i) * (LON2 - LON1)

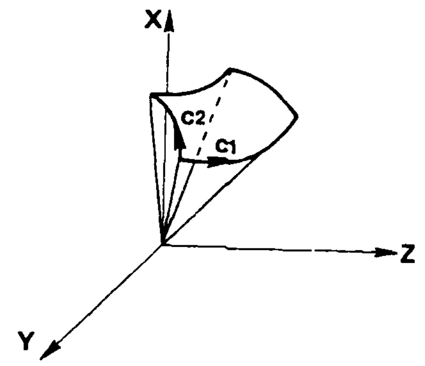
for i=1,2,--,NLON. In each case, the angles will be equally spaced if no distribution is specified.

The routine first converts all angles to radians. If no limiting latitude angles are specified, LAT1 is set to 0, and LAT2 is set to π for the closed surfaces, or to $\pi/2$ for the open ones. Similarly, if no longitude limits are given, LON1 is set to 0, and LON2 is set to 2π in all cases.

The radius is calculated at each point as a function of the two angles from the equations given in Appendix C, and is placed in the real array RAD(NLON,NLAT). The points on the surface are then placed in the real array R(3,NLON,NLAT) according to

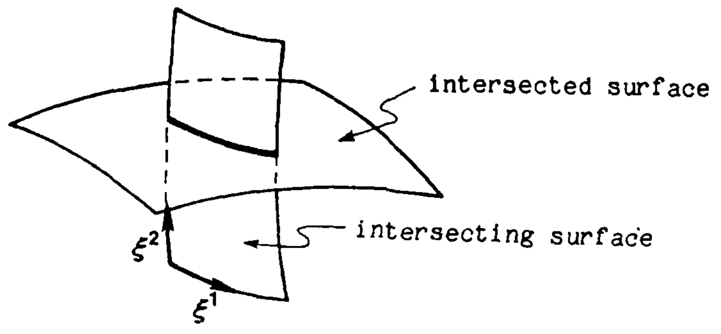
```
                              θ          ψ
                              -          -
R(1,i,j) = RAD(i,j) * SIN(LAP) * COS(LOP)
R(2,i,j) = RAD(i,j) * SIN(LAP) * SIN(LOP)
R(3,i,j) = RAD(i,j) * COS(LAP)
```

for i=1,2,--,NLON and j=1,2,--,NLAT, for return through equivalance with CORD in COMMON/RAY/. Here LOP is calculated for each i, and LAP for each j, as noted above. The first index of the grid on the surface varies on latitude lines, while the second varies on the longitude lines:



16. INTSEC

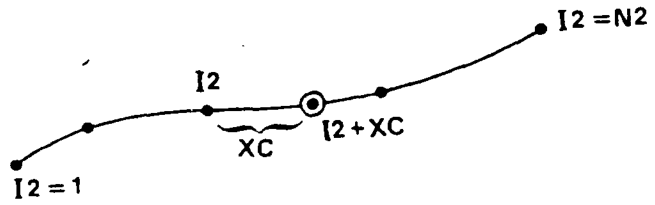
This subroutine generates a curve as the intersection of two surfaces.



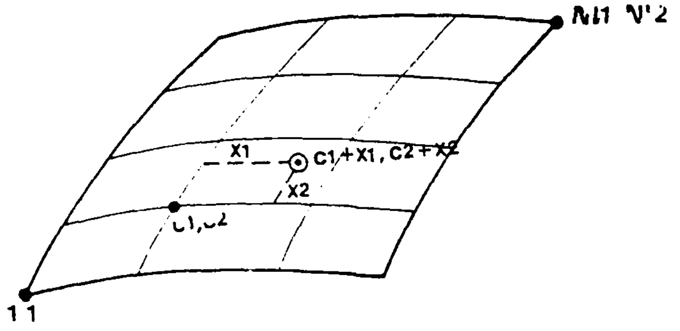
The intersection curve is composed of the intersections of one family of curves, i.e., the curves in the second direction on the N1xN2 intersecting surface with the NI1xNI2 intersected surface. The dimensions of the two surfaces are received through equivalence with POINTS for the intersecting surface, and with POINTSI for the intersected surface, in COMMON/PAR/. The surface spline of the intersected surface is received in SUR(3,NI1,NI2) in COMMON/SPLINE/, and the intersecting surface in CORD(3,N1,N2) in COMMON/RAY/.

Each of the N1 curves (say curve I1) in the second direction on the intersecting surface in succession is splined by copying the curve from CORD(3,I1,N2) to CUR(3,N2,0), and calling CURSPL to return the spline in CUR(3,N2,0:1). This spline has constant-curvature ends. The intersection of this spline curve CUR with the surface spline SUR of the intersected surface is then found by Newton iteration.

Here there are three variables in the iteration: the spline coordinate XC on the curve and the two spline coordinates, X1 and X2, on the surface. Since the splines are with respect to the curvilinear coordinates, i.e., the integer point indices, the actual position on the spline curve is I2+XC, where I2 is the integer point index (I2=1, 2,--,N2) and XC varies on the range 0-1:



Similarly on the surface, the point location is C1+X1,C2+X2 where C1=1,2,--,NI1 and C2=1,2,--NI2 with X1 and X2 on the range 0-1:



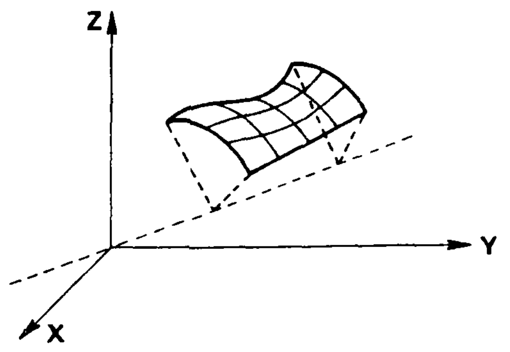
The iteration consists of an inner and an outer iteration, where the inner iteration is Newton iteration for X1,X2,XC on a given interval, i.e., with C1,C2,I2 fixed. If the inner iteraton leaves this interval, the outer iteration changes C1,C2,I2, i.e., proceeds to a different interval.

At each outer iteration the surface spline coefficients for the current C1,C2 are placed in Q(4,4,3) from SUR (Section II-C4), and the curve spline coefficients for the current I2 are placed in QC(4,3) from CUR. The Newton iteration for X1,X2,XC then proceeds on the current interval (Appendix H) until either convergence or one of these three variables leaves the interval, in which case the values of C1,C2,I2 for indicated location are determined, and a new outer iteration occurs.

Upon convergence, the Cartesian coordinates of the intersection are determined from the surface spline and are placed in CORD(3,I1,1). This procedure is followed for each of the curves on the intersecting surface for I1=1,2,--,N1. The intersection curve thus is returned in CORD(3,N1,1) through COMMON/RAY/.

17. ROTATE

This subroutine generates a surface by rotating a space curve about an axis.

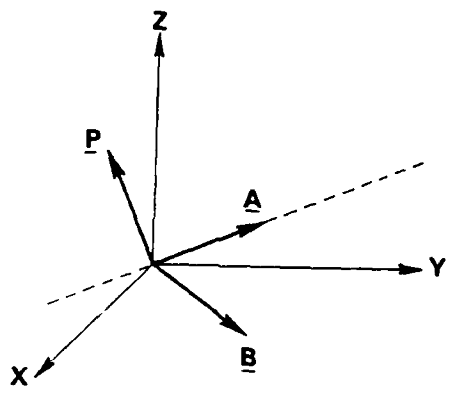


The routine receives two space curves in the same axis system and having the same number of points, N1, as input in the real arrays RI1(3,N1) and RI2(3,N1). If the second curve is not given, it is made the same as the first curve. These curves are received through equivalence with CURV in COMMON/RAY/, and N1 is received through equivalence with POINTS(1).

Also received are the three direction cosines of an axis of rotation passing through the origin. The direction cosines of the rotation axis are received in the array COSINES(k,3) for k=1,2,3, through equivalence with PARAM in COMMON/PAR/. The principal normal to the rotation axis is calculated as the maximum of $\underset{\sim}{i} \times \underset{\sim}{A}$, $\underset{\sim}{j} \times \underset{\sim}{A}$, and $\underset{\sim}{k} \times \underset{\sim}{A}$, in magnitude, normalized as a unit vector. The routine generates the direction cosines of the binormal to the rotation axis from the relation

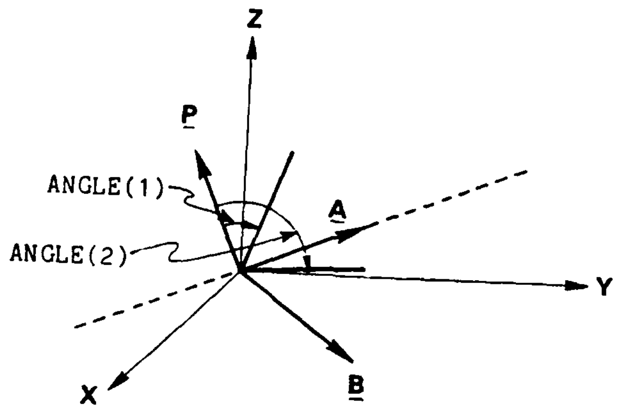
$$\underset{\sim}{B} = \underset{\sim}{A} \times \underset{\sim}{P}$$

where $\underline{A}$,$\underline{P}$, and $\underline{B}$ are the unit vectors in the directions of the rotation axis, its principal normal, and its binormal, respectively:



The direction cosines of the binormal are placed in COSINES(k,2).

Two limiting rotation angles are also received, ANGLE1 and ANGLE2, through equivalence with ANGLE in COMMON/PAR/, and the rotation will be from the former to the latter, either of which may be the larger. These angles are measured from the normal toward the binormal, i.e., clockwise looking down the rotation axis:



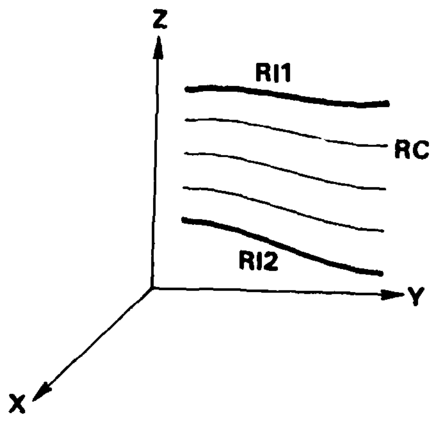
If only one angle is given, the second is set to the first plus 2π, i.e., for a full rotation. The number of rotation angles to be used is received as N2, through equivalence with POINTS(2) in COMMON/PAR/, and a relative angular distribution can be specified in the array ANGDIS(N2), having monotonic values from 0 to 1. This distribution array is equivalenced with DISTRIB(j, 1), for j=1,2,--,N2, in COMMON/RAY/. The rotation angles are set according to

ANGLE1 + ANGDIS(j) * (ANGLE2 - ANGLE1)

for j=1,2,--,N2. If no distribution is given, the angles will be equally spaced.

A set of N2 curves, one for each rotation angle, is generated as follows. For each value of j from 1 to N2, a space curve is generated as an interpolation between the two input curves, using a relative distribution (monotonic values from 0 to 1) in the array CURDIS(N2), received through equivalence with DISTRIB(j,2) in COMMON/RAY:

$$\underline{RC} = \underline{RI1} + CURDIS(j) * (\underline{RI2} - \underline{RI1})$$



A linear distribution is used if none is given. This interpolated curve, $\underline{RC}$, is then rotated by the angle given above, to produce the curve on the surface for this value of j.

The rotation is accomplished as follows at each point on the interpolated curve. The components of the vector to the point in the system formed by the unit vectors $\underline{A}$, $\underline{P}$, and $\underline{B}$ are calculated and placed in the array RR(3) according to the relations

$$\underline{RR} = \begin{bmatrix} \underline{RC} \cdot \underline{P} \\ \underline{RC} \cdot \underline{B} \\ \underline{RC} \cdot \underline{A} \end{bmatrix}$$

where the array RC(3) contains the components of the point vector in the $\underline{i},\underline{j},\underline{k}$ system. The vector to the point can thus be expressed as

$$RR(1)\underline{P} + RR(2)\underline{B} + RR(3)\underline{A}$$

The $\underline{P}$, $\underline{B}$, $\underline{A}$ system then is rotated about $\underline{A}$ through the angle ANG to form a new system $\underline{P}'$, $\underline{B}'$, $\underline{A}'$ according to

$$\begin{bmatrix} \underline{P}' \\ \underline{B}' \\ \underline{A}' \end{bmatrix} = \begin{bmatrix} \underline{P}' \cdot \underline{P} & \underline{P}' \cdot \underline{B} & \underline{P}' \cdot \underline{A} \\ \underline{B}' \cdot \underline{P} & \underline{B}' \cdot \underline{B} & \underline{B}' \cdot \underline{A} \\ \underline{A}' \cdot \underline{P} & \underline{A}' \cdot \underline{B} & \underline{A}' \cdot \underline{A} \end{bmatrix} \begin{bmatrix} \underline{P} \\ \underline{B} \\ \underline{A} \end{bmatrix}$$

or

$$\begin{bmatrix} \underline{P}' \\ \underline{B}' \\ \underline{A}' \end{bmatrix} = \begin{bmatrix} \cos ANG & -\sin ANG & 0 & \underline{P} \\ \sin ANG & \cos ANG & 0 & \underline{B} \\ 0 & 0 & 1 & \underline{A} \end{bmatrix}$$

The components of the rotated point vector in the $\underline{P}$, $\underline{B}$, $\underline{A}$ system then are given by

$$\begin{bmatrix} RC(1) \\ RC(2) \\ RC(3) \end{bmatrix} = \begin{bmatrix} \cos ANG & -\sin ANG & 0 & RR(1) \\ \sin ANG & \cos ANG & 0 & RR(2) \\ 0 & 0 & 1 & RR(3) \end{bmatrix}$$

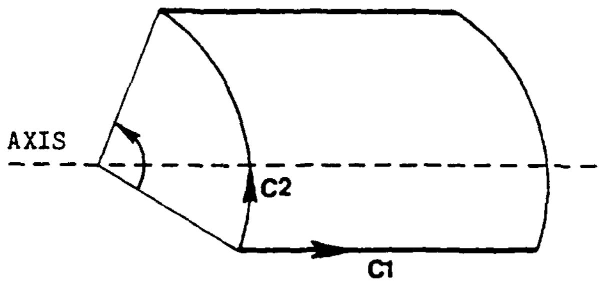
The rotated point vector expressed in the $\underline{P}$, $\underline{B}$, $\underline{A}$ system then is

$$RC(1)\underline{P} + RC(2)\underline{B} + RC(3)\underline{A}$$

and the components in the $\underline{i},\underline{j},\underline{k}$ system are thus given by

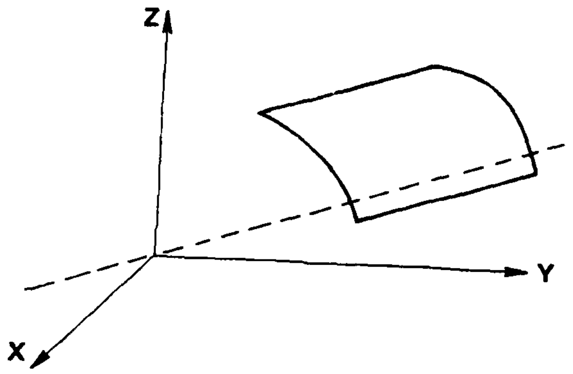
$$\underline{R} = \begin{bmatrix} RC(1)(\underline{P} \cdot \underline{i}) + RC(2)(\underline{B} \cdot \underline{i}) + RC(3)(\underline{A} \cdot \underline{i}) \\ RC(1)(\underline{P} \cdot \underline{j}) + RC(2)(\underline{B} \cdot \underline{j}) + RC(3)(\underline{A} \cdot \underline{j}) \\ RC(1)(\underline{P} \cdot \underline{k}) + RC(2)(\underline{B} \cdot \underline{k}) + RC(3)(\underline{A} \cdot \underline{k}) \end{bmatrix}$$

in the array R(3, i,j) for i=1,2,--,N1 for return through equivalence with CORD in COMMON/RAY/. The first index of the grid on the resulting surface varies along the curves, while the second varies around the rotation axis:



18. STACK

This subroutine generates a surface by stacking space curves along a space curve axis:

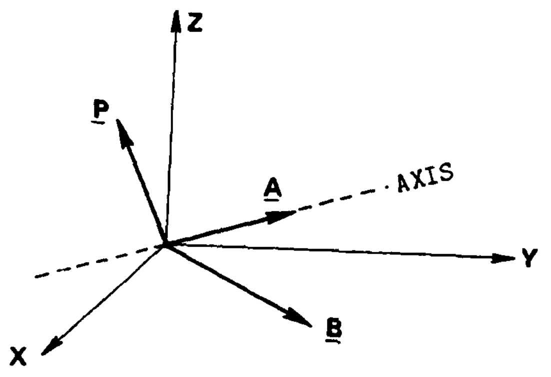


The routine receives two space curves in the same axes system and having the same number of points, N1, in the real arrays RI1(3,N1) and RI2(3,N1). These two curves are received through equivalence with CURV in COMMON/RAY/, and N1 is equivalenced with POINTS(1) in COMMON/PAR/. If the second curve is not given, it is made the same as the first curve. The space curve axis, with N2 points, is received in the array RO(3,N2). The unit tangent and principal normal at each point on this axis are generated elsewhere and are received here in the real arrays TNG(3,N2) and NOR(3,N2), respectively. The axis, and its tangent and normal, are received through equivalence with AXIS, TANGEN1, and NORMAL in COMMON/RAY/, and N2 through equivalence with POINTS(2) in COMMON/PAR/.

Also received are the three direction cosines of the axis, and the direction cosines of the principal normal to this axis, in the system in which the curves are defined. The direction cosines of the axis are received in the array COSINES(k,3) for k=1,2,3, and those of the principal normal are in COSINES(k,1). These arrive through equivalence with PARAM in COMMON/PAR/. The routine generates the direction cosines of the binormal to the axis in the curve system from the relation

$$\underset{\sim}{B} = \underset{\sim}{A} \times \underset{\sim}{P}$$

where $\underset{\sim}{A}$, $\underset{\sim}{P}$, and $\underset{\sim}{B}$ are the unit tangent, principal normal, and binormal, respectively, to the axis:



The direction cosines of the binormal are placed in COSINES(k,2).

A set of N2 curves, one for each point on the axis is generated as follows. At each point on the axis, i.e., for j=1,2,--,N2, the local binormal to the axis is calculated and placed in the array BIN(3) according to

$$\underline{BIN} = \underline{TNG} \times \underline{NOR}$$

A space curve is generated as an interpolation between the two input curves, using a relative distribution (monotonic values from 0 to 1) in the array CURDIS(N2), equivalenced with DISTRIB(J, 2) in COMMON/RAY/ for j=1,2,--,N2:

$$\underline{RC} = \underline{RI1} + CURDIS(j) * (\underline{RI2} - \underline{RI1}) \quad \text{(cf. Section II-D17)}$$

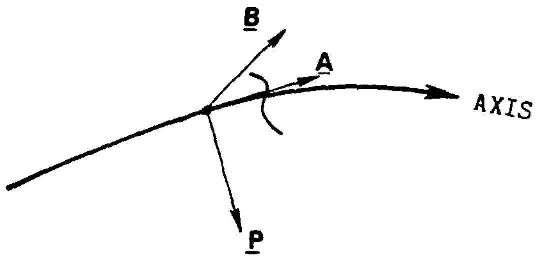
A linear distribution is used if none is given. At each point on the interpolated curve the components in the system formed by the unit vectors $\underline{A}$, $\underline{P}$, and $\underline{B}$ are calculated and placed in the array RR(3) according to the relations

$$\underline{RR} = \begin{pmatrix} \underline{RC} \cdot \underline{P} \\ \underline{RC} \cdot \underline{B} \\ \underline{RC} \cdot \underline{A} \end{pmatrix}$$

where the array RC(3) contains the components of the point vector in the $\underline{i},\underline{j},\underline{k}$ system. The vector to the point can then be expressed as

$$RR(1)\underline{P} + RR(2)\underline{B} + RR(3)\underline{A}$$

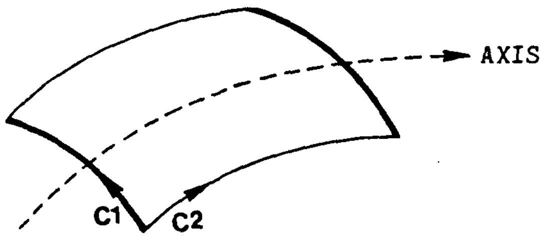
The $\underline{P}$, $\underline{B}$, $\underline{A}$ system is then moved to have its origin at the point $\underline{RO}(j)$ on the axis, with $\underline{A}$ aligned with the local tangent to the axis, $\underline{P}$ aligned with the local principal normal, and $\underline{B}$ aligned with the local binormal:



The components of the moved point vector in the $\underline{i},\underline{j},\underline{k}$ system are thus given by

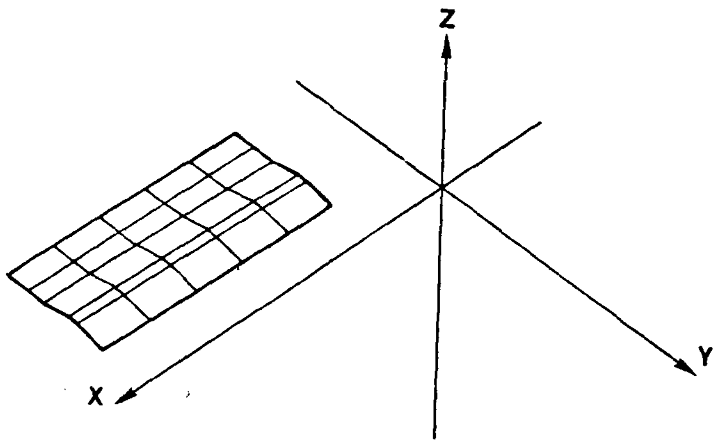
$$\underline{R} = \underline{RO} + RR(1)\underline{NOR} + RR(2)\underline{BIN} + RR(3)\underline{TNG}$$

The points on the moved curve are placed in the array R(3,i,j) for i=1,2,--,N1 for return through equivalence with CORD in COMMON/RAY/. The first index of the grid on the resulting surface varies along the curves, while the second varies down the axis:



19. BLEND

This subroutine generates a surface by interpolation between two space curves.

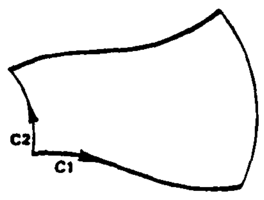


It receives two space curves in the same axes system having the same number of points, N1, in the real arrays RI1(3,N1) and RI2(3,N1). These curves are received through equivalence with CURV in COMMON/RAY/, and N1 through equivalence with POINTS(1) in COMMON/PAR/.

A set of N2 space curves is generated by interpolation between these two input curves using a relative distribution (monotonic values from 0 to 1) in the array CURDIS(N2), equivalenced with DISTRIB(j,2) in COMMON/RAY/ for j=1,2,--,N2.

$$\underline{R} = \underline{RI1} + CURDIS(j) * (\underline{RI2} - \underline{RI1})$$

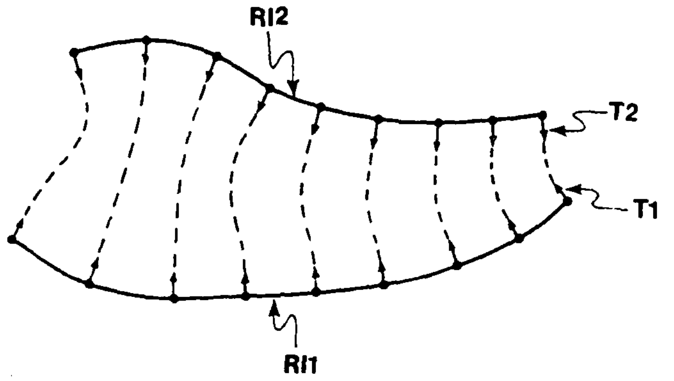
for j=1,2,--,N2. A linear distribution is used if none is specified. The points on the interpolated curve are placed in the array R(3,i,j) for i=1,2,--N1 for return through equivalence with CORD in COMMON/RAY/: The first index of the grid on the resulting surface varies along the curves, while the second varies from one interpolated curve to the next:



20. PATCH

This subroutine generates a surface by connecting two space curves by a set of cubic curves. The routine receives two space curves having the same number of points, N2, in the arrays RI1(3,N2) and RI2(3,N2). Also received are tab vectors (not necessarily unit vectors) for each point on each curve, these being in the arrays T1(3,N2) and T2(3,N2). The curves and tab vectors are received through equivalence with CURV and TABVEC in COMMON/RAY/, and N2 is received through equivalence with POINTS(2) in COMMON/PAR/.

A set of N2 cubic curves is generated connecting corresponding points on the two input curves, and matching the tab vectors at each point on the input curves. The number of points on each of these connecting curves is specified as N1, equivalenced with POINTS(1), in COMMON/PAR/:



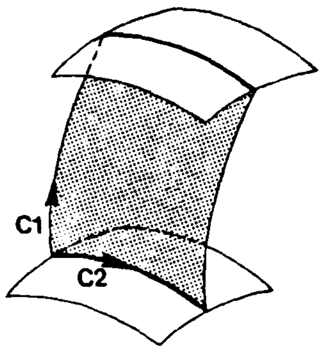
If the spacing in SPACE(_,1), where the first subscript is 1 or 2 referring to the first or second tab curve, is not zero then the corresponding tab vectors are normalized to unity and then multiplied by that spacing. (If TOTARC is not zero the spacing is taken to be relative and the spacing is multiplied by TOTARC.) Zero spacing implies that the spacing is to be the magnitude of the tab vector as received. For each point on the input curves, i.e., for j=1,2,--N2, the routine first calculates the vectors $\underline{r}_{\underline{\xi}}$, where $\underline{\xi}$ represents the normalized coordinates (range 0-1) varying along the connecting curves, from the two sets of tab vectors, placing these derivatives in the arrays S1(3) and S2(3) for the two curves:

$$S1 = (N1 - 1) * T1$$

$$S2 = (N1 - 1) * T2$$

The coefficients in the cubic polynomial are then calculated from the relations given in Appendix D, and N1 points are generated on each connecting curve using equally spaced increments in $\underline{\xi}$.

The point distribution on the connecting curve is then changed to a hyperbolic function distribution (Section I-B6) while preserving the present spacing at each end as follows. First the present spacing at each end is calculated, and is converted to relative spacing by division by the total arc length of the curve, the latter being obtained by a call of subroutine ARCLNGT. Then subroutine RELDIST is called to provide a hyperbolic function relative distribution having the spacing as determined at each end. Subroutine CURDIST is then called to redistribute the points on the connecting curve. If the opration is not on a curved surface, SPLNSUR is called to spline the connecting curve in RI (Section II-C4). Finally, CORPAR is called to convert the Cartesian coordinates in R to spline coordinates of the closest points on the spline, and PARCOR is then called to convert these spline coordinates back to Cartesian coordinates of the final points on the connecting curve. (This is done because the points from CURDIST are set by linear interpolation on chords between points on the cubic curve, and therefore must be transferred onto the cubic curve via the spline.) The revised point distribution on the connecting curve is placed in the array R(3,i,j) for i=1,2,--,N1 for return through equivalence with CORD in COMMON/RAY/. The first index of the grid on the resulting surface varies on the connecting curves, while the second varies on the input curves:

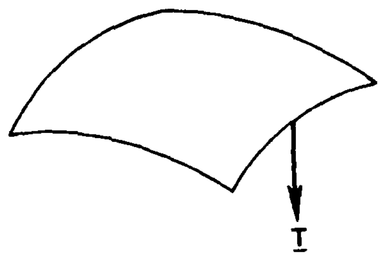


21. EDGETAB

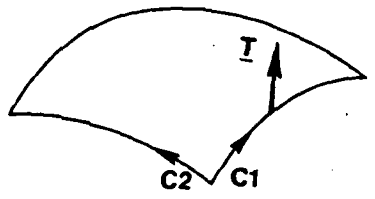
This subroutine receives an N1xN2 surface in the array RI(3,N1,N2), and returns one of the four edges of the surface in the array R(3,N), together with a set of slope vectors in T(3,N), to be used to join another surface to the given surface along the chosen edge with intersection slope determined by the vectors in T. These slope vectors are not necessarily unit vectors but may reflect the spacing on the surface. The number of points on the edge, N, will be either N1 or N2, of course. The input surface is received through equivalence with CORDI in COMMON/RAY/, and its dimensions through equivalence with POINTS in COMMON/PAR/.

The vectors in T may be tangents to the surface off its edge, or may be normals to the surface, as specified by the following values of TABTYP, received in COMMON/PAR/:

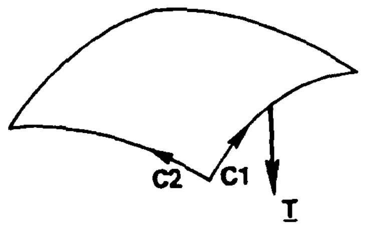
"TANGEN" - tangent to the surface, directed off the edge



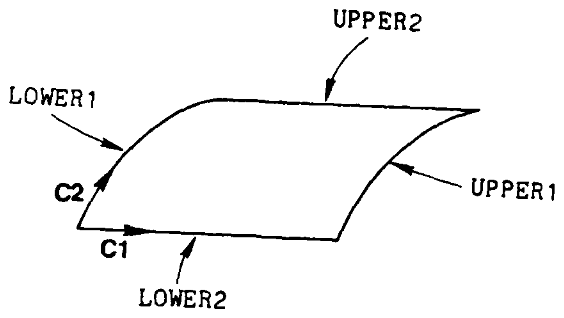
"NORPOS" - positive unit normal to the surface



"NORNEG" - negative unit normal to the surface

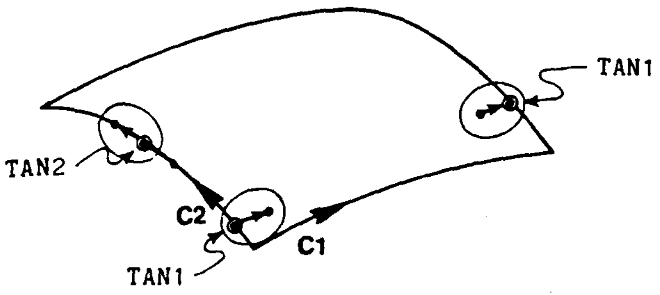


The particular edge to be taken is identified by the value of EDGE, received in COMMON/PAR/:



If the surface is actually only a curve in the x-y plane, two parallel curves are added at unit separation in the z-direction to form a surface so that no special procedures will be needed.

On the specified edge, the routine determines the tangents to the two sets of curves on the surface using two-point central differences between points on the edge and two-point one-sided differences off the edge as illustrated below:



These tangents are not unit vectors but rather reflect the spacing on the surface. The normal is then determined from

<u>NOR</u> = <u>TAN1</u> x <u>TAN2</u>

and is normalized to a unit vector. The appropriate tangent, or the normal or its negative, are placed in the slope vector array T, and the points on the edge are placed in the array R.

22. INTERP1, INTERP2, and INTERP3

These subroutines perform linear interpolation for one, two, or three functions with NC components. All transfers here are through the arguments. The routines receive arrays of NI function values in FI(NC,NI) for INTERP1, in FI1(NC,NI) and FI2(NC,NI) for INTERP2, and in these and FI3(NC,NI) for INTERP3. The several functions in the latter two routines are not related. An array of NI arc length values, corresponding to the function values, is received in the array AI(NI). Finally an array of NO values of arc length at which the function(s) is to be evaluated by the interpolation is received in the array AO(NO).

The routines evaluate the function(s) at the arc lengths in AO by linear interpolation among the function values in FI(or FI1, etc.) corresponding to the arc lengths in AI. The interpolated function values are placed in the array FO(NC,NO) for INTERP1 and in FO1(NC,NO), etc., for the others.

23. LINE

This subroutine generates a straight line between two end points specified by the arrays R1(3) and R2(3), received in COMMON/PAR/, with N points on the line, N being equivalenced with POINTS in COMMON/PAR/. A relative distribution, with monotonic variation from 0 to 1, can be given in the array FRAC(N), equivalenced with DISTRIB(i,1) for i=1,2,--,N in COMMON/RAY/. The points are generated according

$$\underline{R1} + FRAC(i) * (\underline{R2} - \underline{R1})$$

for i=1,2,--,N and are placed in the array R(3,N), for return through equivalence with CORD in COMMON/RAY/.

24. PUT

This subroutine places a surface(curve) on file or core storage. It also prints and/or plots a surface(curve). The N1xN2 surface(curve) is received in the argument array F(3,N1,N2), with N2=1 for curves. The surface dimensions, N1 and N2, are also received as arguments. An optional label is received in the integer LABOUT in COMMON/IO/, which is included in printed notations and on plots but is not stored.

Core Storage

Storage on core is called for by a positive value of the integer COROUT in COMMON/IO/, the value indicating the storage number. Core storage is in the large array STORE(3,DIMSS) in COMMON/IO/. The same number cannot, of course, be used for two surfaces without overwriting the one already on storage.

Both surfaces and curves are stored one-dimensionally in STORE as sequences of points (three Cartesian coordinates for each point, with the first surface dimension running faster). The maximum number of points that can be put in STORE is DIMSS. If the present surface will not fit above those already in STORE, the array STORE is written on file

10 and a new STORE array is opened. In this manner a number of successive writes of STORE to file may occur, and no limit is imposed as to the total number. Each successive STORE on the file is numbered consecutively from 1. The contents of a certain STORE are retrieved from the file by reading through all preceding versions on the file. (It is thus desirable to have DIMSS as large as possible.)

Once a version of STORE is written on file 10, no changes are ever made in that version. The total number of versions on file 10 is kept in MSTORE. A copy of a partially-filled version of STORE is kept on file 7, and PSTORE=1 indicates its existence. The version of STORE on file 10 which contains a certain stored surface, i, is recorded in NSTOR(i), and the location of the first point on this surface in STORE is recorded in LSTOR(i). (The maximum number of surfaces that can be stored in this manner is DCOR.) The number of the version of STORE that is currently in core is kept in NSTORE. File 10 is not rewound after being written or read. If NSTORE is less than MSTORE then the routine first reads file 10 from its present position until the last version of STORE on the file is in core. If NSTORE is greater than MSTORE, i.e., if the version of STORE currently in core is the partially-filled version, then the routine will attempt to add the present suface to this version in core. Otherwise the version currently in core is not the partially-filled version, and the partially-filled version is recovered from file 7 after rewinding.

The starting position for the next storage in this last version of STORE is kept in LSTORE. Therefore, if LSTORE+N1*N2-1 exceeds DIMSS, the present surface to be stored will not fit in this version of STORE.

In that case a new version of STORE is opened by incrementing MSTORE and setting LSTORE to 1 and PSTORE to 0. In any case NSTORE is set to MSTORE+1 and the present surface is written in STORE starting at position LSTORE, and the STORE version number is placed in NSTOR(COROUT) and the starting point location in LSTOR(COROUT). Finally, LSTORE is incremented by N1*N2.

Note that, although core storage numbers may be reused, the space in STORE is not released. Therefore no economy is gained by reusing storage numbers.

The routine stores the dimensions of the surface, N1 and N2, in CORES(1,COROUT) and CORES(2,COROUT), respectively, in COMMON/IO/. The array F is written into the array STORE as described, and a notation of the storage is printed.

File Storage

Storage on file is called for by a positive value of the integer FILOUT in COMMON/IO/. The array F is written onto a disk file by a loop with the first subscript running fastest, followed by the second and the third. The format on file storage is controlled by TRIAD and FORM (Section I-B2), received as LTRIAD and LFORM through COMMON/IO/. If HEAD (received as LHEAD through COMMON/IO/) is "YES", a counter and the surface dimensions N1,N2 are written as a triad, also in the format indicated by FORM, before each surface on the file. The form is unformatted if LFORM in COMMON/IO/ is eqal to "NFORM", E20.8 format if "E", or list-directed if "LIST". The file is rewound before being written on unless REWOUT="NO" in COMMON/IO/. The system file number is FILOUT+10, the 10 being added to avoid system files. This allows the file storage locations specified by FILOUT to start with 1, but must be taken account of in runstream statements that preserve these files, e.g., the file created with FILOUT=1 must be preserved by a runstream statement referring to file 11.

A file storage number, FILOUT, must not be used again unless overwriting is intended. No limit on the maximum number of files is imposed by the code, but there will be some system limit, of course, and the code cannot check for violations. The routine stores the dimensions of the surface(curve), N1 and N2, in FILES(1,FILOUT) and FILES(2,FILOUT), respectively, in COMMON/IO/, and prints a notation of the storage.

Print

If either entry of the integer array OUT(2) in COMMON/IO/ is equal to "PRINT", the surface(curve) in the array F will be printed.

Plot

If either entry of the integer array OUT(2) is equal to "PLOT", the surface(curve) in the array F will be plotted. If FRAME="NEW" the plot will be on a new frame; otherwise it is added to the previous frame to form a composite plot. The routine numbers the frames in NPLOT, and calls the system subroutine ENDPL, with NPLOT as the argument, if a new frame is indicated.

The limits of the plot are received in the arrays RMIN(3) and RMAX(3), and if none are given, and FRAME="NEW", they are set to the limits of the values in the array F, i.e., the entire surface(curve) is plotted. These plot limits are printed. The viewpoint for the plot is received in the array VIEW(3), the first two entries of which are the viewing angles (Section I-B5) and the third is the distance. This distance is set to 1000 times the diagonal of the box formed by the corners defined by RMIN and RMAX if no third entry is given for VIEW. The two view angles are defaulted in the main program to 90°. The physical size of the plot on the screen is specified by the array SIZE(2), the entries being the horizontal and vertical dimensions in inches. These dimensions are defaulted in the main program to 8 inches. All of these plot parameters are received in COMMON/IO/.

The plot is set up by calling the system subroutines TITL3D (to include the label and to set the screen size), AXES3D (to draw axes), VUANGL (to set the viewpoint), and GRAF3D (to set the plot limits). Each family of grid lines on the surface is then plotted by placing the three Cartesian coordinates of each point on each line in succession in the three arrays XRAY, YRAY, ZRAY (the dimensions of which are N1 or N2, as the case may be) and calling the system subroutine CURV3D to draw the line.

25. <u>GET</u>

This subroutine retrieves a surface or curve from file or core storage, reads it from the namelist, or transfers it from the current array CORD. The N1xN2 surface (or curve with N2=1) is returned in the argument array F(3,N1,N2,). An optional label is received in the integer LABIN in COMMON/IO/, which is included in printed notation but is not used for location identification.

From Core

Retrieval from core storage is called for by a positive value of the integer CORIN in COMMON/IO/, this value being the same as used previously for COROUT when the storage was done. The routine obtains the dimensions of the surface, N1 and N2, from CORES(1,CORIN) and CORES (2,CORIN) in COMMON/IO/. The routine obtains the number of the STORE version on file 10 containing the present surface from NSTOR(CORIN) and retrieves this version from the file. If this version number is greater than the number NSTORE of the version currently in core then file 10 is read from its current position through the version needed. If the version needed is the partially-filled version then this version is read from file 7 after rewinding. If, however, the number of the version needed is less than the number of the version currently in core, then if the partially-filled version is in core it is written on file 7 after rewinding and PSTORE is set to 1. Then file 10 is rewound and read through the version needed.

The surface is then written from STORE, starting at location LSTOR (CORIN), to the argument array F, and a notation of the retrieval is printed.

From file

Retrieval from file storage is called for by a non-zero value of the entry FILIN in COMMON/IO/. A positive value of FILIN indicates that the retrieval is to be from a file established by subroutine PUT, with FILIN having the same value as used previously for FILOUT when the storage was done. A negative value of FILIN indicates that the surface(curve) was written on file by the grid code. In either case the file is rewound before being read unless REWIN="NO" in COMMON/IO/. Recall that the actual disk file number is |FILIN| + 10, i.e., FILIN=1 means disk file 11. This must be considered in runstream statements that obtain the disk files.

If FILIN is positive, the routines obtain the dimensions of the surface(curve), N1 and N2, from FILES(1,FILIN) and FILES(2,FILIN) in COMMON/IO/ unless the former is zero in which case the argument values are used. The array F is read from disk file number FILIN+10 in a loop with the first subscript of F running fastest, followed by the second and the third. The format on a file read is indicated by TRIAD and FORM (Section I-B2), received as LTRIAD and LFORM through COMMON/IO/. The form is unformatted if LFORM in COMMON/IO/, is equal to "UNFORM", E20.8 format if "E", or list-directed if "LIST".

If FILIN is negative, the routine reads from disk file number |FILIN| + 10, reading first the number of blocks into the integer BMAX, and then the dimensions of each block into the integer array CMAX (3,BMAX), using an implied loop with the first subscript running faster. For each block, the values of F are read using an implied loop on the first subscript in an outer loop with the second subscript running faster. Subroutine PUT is then called to store the surface(curve) for that block on file (or core) at successive locations starting with that received in FILOUT (or COROUT). After all of the blocks have been stored, COROUT and FILOUT are set to zero. A notation of the retrieval is printed in either case.

From namelist

If the first entry of the array VALUES in COMMON/IO/ is not equal to "NONE" (the default), then the routine reads N1xN2 points from this array in the namelist into the argument array F(3,N1,N2). This reading is done in a loop with the first subscript in F varying fastest, followed by the second and third. The maximum number of points that can be read for a surface is DIMV, and an error check is made. A notation of the reading is printed.

From the current array

If both CORIN and FILIN are zero, and the first entry of the array VALUES is "NONE" (the defaults for each), the argument array F(3,N1,N2) is simply copied from the current surface array CORD(3,N1,N2) in COMMON/RAY/ and a notation is printed. The dimensions here are obtained from the values, NC1 and NC2, stored for the current surface.

26. SWITCH

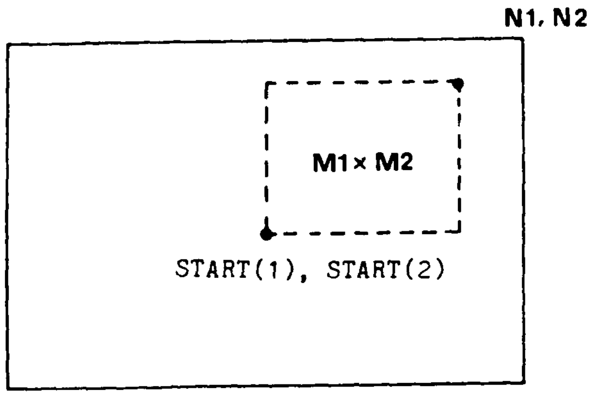
This subroutine switches the order of progression of the points in either or both directions on a surface or curve, and/or switches the grid point indices on a surface. The N1xN2 surface(curve) is received in the array R(3,N1,N2), and the switching operation is indicated by the integer array REORDER(3), equivalenced with START in COMMON/PAR/. The surface array is received through equivalence with CORD in COMMON/RAY/, and its dimensions are received through equivalence with POINTS in COMMON/PAR/.

If any entry of REORDER is equal to "REVERSE1", then the progression of the second index in R is reversed, i.e., the value in R supplied for the second index equal to N1 will be returned in R for the second index equal to 1, etc. A similar effect is produced with regard to the third index if any entry of REORDER is equal to "REVERSE2". If any entry is equal to "SWITCH" then the entries in R will be rearranged so that the order of the second and third subscripts is switched, i.e., R is returned in the form R(3,N2,N1). Any of these operations can be called for simultaneously, and reversal of progression is performed before switching.

The routine accomplishes this by first copying the array R as received to the array RI(3,N1,N2). Then RI(3,i,j) is copied back to R(3,m,n) in a loop. In this loop, i varies from N1 to 1 if "REVERSE1" is found, or from 1 to N1 otherwise. Similarly, j varies from N2 to 1 if "REVERSE2" is present, or from 1 to N2 otherwise. If "SWITCH" is found, m=j and n=i, thus accomplishing the switch; otherwise m=i and n=j.

27. INSERT

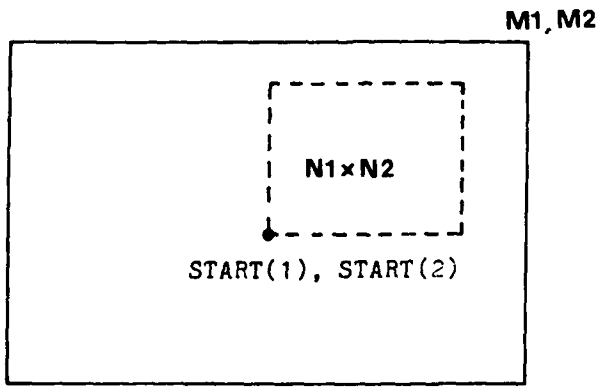
This subroutine inserts the M1xM2 argument array RI(3,M1,M2) into the N1xN2 argument array R(3,N1,N2), starting at the position specified by the array START(2):



This is done in a loop that copies RI(3,i,j) onto R(3,m,n) with m=START(1)+i-1 and n=START(2)+j-1 for i=1,2,---, M1 and j=1,2,--, M2. The surface dimensions, M1 and M2 are equivalenced with POINTSI in COMMON/PAR/, and N1 and N2 are equivalenced with POINTS.

28. <u>EXTRACT</u>

This subroutine extracts an N1xN2 argument array R(3,N1,N2) from an M1xM2 array RI(3,M1,M2), starting at the position specified by the array START(2):



This is done in a loop that copies RI(3,i,j) onto R(3,m,n) with i=START(1)+m-1 and j=START(2)+n-1 for m=1,2,--,N1 and j=1,2,--,N2. The surface dimensions, M1 and M2, are equivalenced with POINTS in COMMON/PAR/, and N1 and N2 are equivalenced with POINTS.

29. SCURVE

This subroutine generates a cubic space curve between two points, with specified slope vectors at each end. The end points are received in the arrays R1(3) and R2(3) in COMMON /PAR/, and the unit tangents at the ends are received in the arrays T1(3) and T2(3), equivalenced with TABVEC in COMMON/RAY/. The number of points on the curve is specified by N, which is equivalenced with POINTS in COMMON/PAR/.

The routine calculates the vectors $\underline{r}_{\underline{\xi}}$, where $\underline{\xi}$ represents the normalized coordinate (range 0-1) varying along the curve, from the vectors in T1 and T2, placing these derivatives in the arrays S1(3) and S2(3):

$$S1 = (N1-1) * T1$$

$$S2 = (N1-1) * T2$$

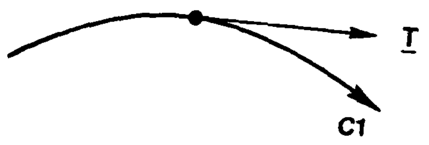
The unit tangents in T1 and T2 here are multiplied by the spacings in SPACE(1,1) and SPACE(2,1). These spacings are multiplied by the total arc length, TOTARC, if they are relative. The parameters of the cubic polynomial are then evaluated from the equations in Appendix D. The cubic curve is generated using a linear distribution of the coordinate $\underline{\xi}$ that varies along the curve from 0 to 1 as the point index varies from 1 to N. The line is placed in the array R(3,N) for return through equivalence with CORD in COMMON/RAY/.

30. CURTAB

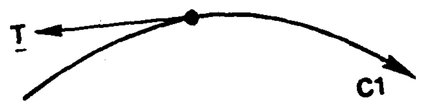
This subroutine receives a curve with N points in the array RI(3,N), and returns a single point, specified by I, on the curve in the argument array R(3), together with a slope vector at the point in the argument array T(3), to be used to join another curve intersecting the given curve at the point R with slope vector T. The curve is received through equivalence with CORDI in COMMON/RAY/, and N and I through equivalence with POINTS and START in COMMON/PAR/.

The vector T may be the tangent or unit principal normal to the curve, as determined by the following values of the integer TABTYP in COMMON/PAR/:

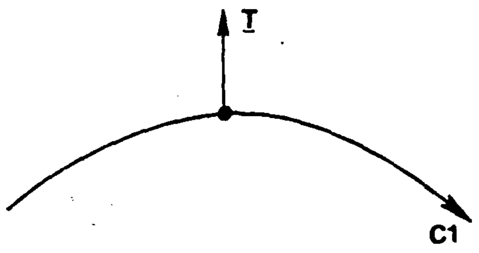
"TANPOS": positive tangent



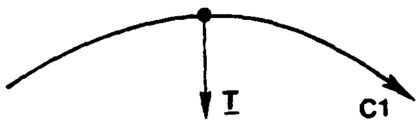
"TANNEG": negative unit tangent



"NORPOS": positive unit normal



"NORNEG": negative unit normal



The closed surfaces are centered on the origin. The ellipsoid has its axes on the three coordinate axes, and the cone and paraboloid have the axis on the z-axes, with the vertex on the positive z-axes.

The routine first calls subroutine CURVEC to determine the tangent and principal normal at each point on the curve. The point at I and the appropriate slope vector there are then placed in R and T.

31. <u>FACDIST</u>

This subroutine receives a set of NI arc lengths in the array SCALE(1,NI) and a corresponding set of NI distribution factors in the array SCALE(2,N). Also received is a set of N arc lengths in the array ARC(N). The routine converts the arc lengths in SCALE(1,NI) to the range of those in ARC by multiplying those in SCALE by the ratio of the last arc length in ARC to the difference between the first and last values in SCALE, placing the result in the array AI(NI). Similarly, the distribution factors in SCALE(2,NI) are converted to relative values on the range 0-1 by division by the difference between the first and last values in SCALE, placing the result in AO(NI).

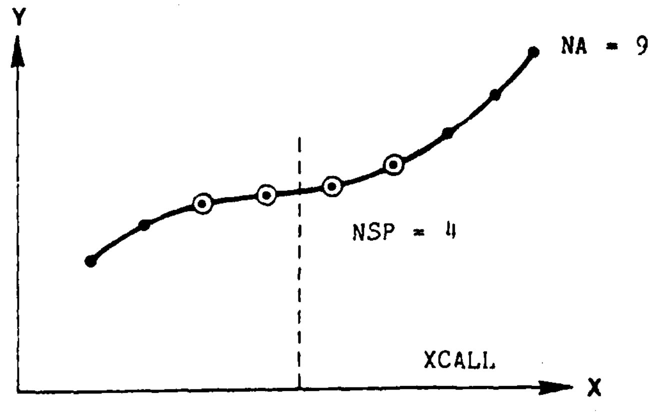
A set of N distribution factors corresponding to the arc lengths received in ARC is then generated from the factors in AO by linear interpolation among the arc lengths in AI. These factors are placed in the array FRAC(N).

32. SPLINE

This subroutine fits a cubic spline to a space curve of NSP points from the NA points in the argument arrays XARAY(NA) and YARAY(NA) defining a curve y(x). The spline span is located about the argument XCALL in the range of values in the array XARAY, centered if possible. The routine returns y, y_x, and y_{xx} at the NSP spline points as the arguments YR, DYDX, and D2YDX2.

The routine first checks that XCALL is in the range of values in XARAY and that the number of spline points, NSP, does not exceed the number of points, NA, in the function arrays. Error messages and termination follow if either of these conditions is not met.

The spline span is then positioned about XCALL in the independent variable array XARAY, centered if possible or against one end of XARAY otherwise:



The NSP points in the span are then splined, using quadratic end spans, through a tridiagonal solution (Appendix G). The function values, and the first and second derivatives, are then calculated at each point on the spline span.

33. SPARC

This routine receives an NPTS-point curve in the array R(3,NPTS) and distributes NMAX points on the curve according to curvature in the array W(3,NMAX). The curve is received through equivalence with CORDI in COMMON/RAY/, and its number of points through equivalence with POINTSI in COMMON/PAR/. The return of the curve is through equivalence with CORD in COMMON/RAY/, the number of points having been received by equivalence with POINTS in COMMON/PAR/.

The routine first calculates the arc length (actually chord length) distribution on the input curve by

$$s(i) = s(i-1) + |\underset{\sim}{R}_i - \underset{\sim}{R}_{i-1}|$$

for i=2,3,--,NPTS. Subroutine SPLINE is then called to spline the curve, providing the first and second derivatives of R with respect to arc length. The radius of curvature is then calculated at each point on the curve from the relation (Appendix E)

$$\rho = \frac{|\underset{\sim}{r}_s|^2}{|\underset{\sim}{r}_s \times \underset{\sim}{r}_{ss}|}$$

The radius of curvature is limited by RHOMX, set to 1000.

Next the derivative of included angle with respect to arc length, x_s, is computed at each point on the curve as the ratio of arc length increment to radius of curvature:

$$\alpha_s = \frac{|\underset{\sim}{r}_s|}{\rho}$$

and a cummulative angle distribution is obtained by integrating these increments along the curve:

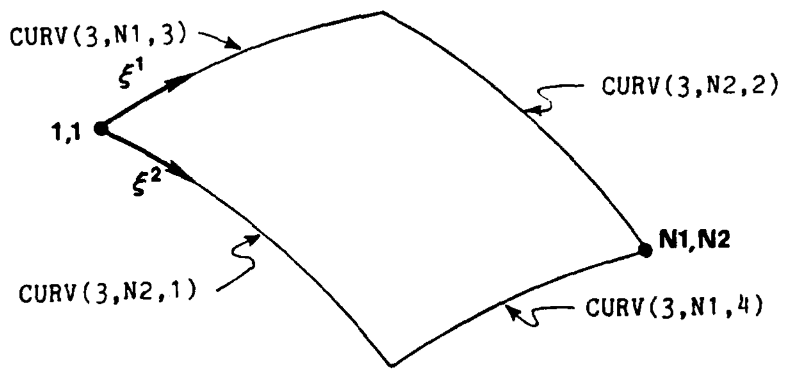
$$\alpha = \int \alpha_s \, ds$$

This angle distribution is then normalized by division by the total accumulated angle.

The NMAX points are then placed in the array W(3,NMAX) on the curve at equal increments of accumulated angle by interpolation on the spline, and are returned through equivalence with CORD in COMMON/RAY/.

34. TRANSUR

This subroutine generates an N1xN2 point surface by transfinite interpolation (cf. Ref. 1 and 2) from four edge curves. The routine receives the four edge curves in the array CURV as follows:



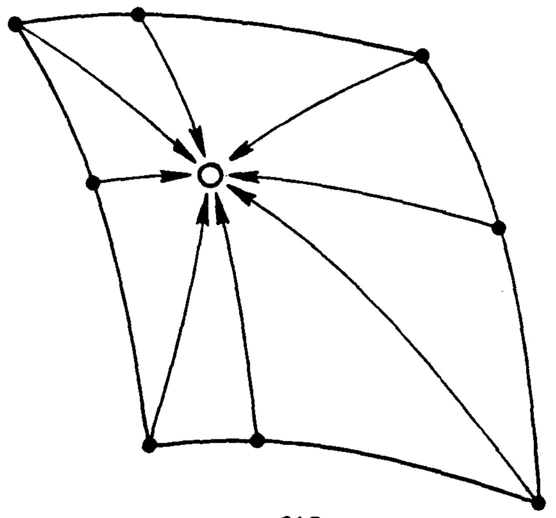
The surface dimensions are received as N1,N2 through equivalence with POINTS(2) in COMMON/PAR/.

The edge curves are first placed appropriately in the array R(3,N1,N2), equivalenced with CORD in COMMON/RAY/. The points on the surface then are generated in R by two-dimensional transfinite interpolation according to

$$\underset{\sim}{r} = P_1 + P_2 = P_1 + P_2 - P_1P_2$$

where P_1 and P_2 are the one-dimensional interpolation projectors in the two directions on the surface (cf. Appendix A of Vol. III). This is implemented as

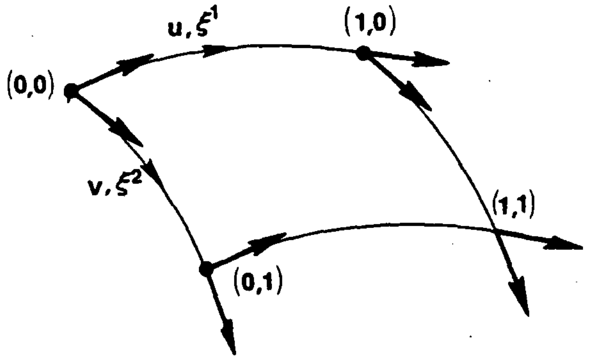
```
R(_,C1,C2) = (1 - F1) * R(_,1,C2)
                  + F1 * R(_,N1,C2)
            + (1 - F2) * R(_,C1,1)
                  + F2 * R(_,C1,N2)
 - (1 - F1) * (1 - F2) * R(_,1,1)
      - (1 - F1) * F2 * R(_,1,N2)
      - F1 * (1 - F2) * R(_,N1,1)
            - F1 * F2 * R(_,N1,N2)
```



35. TENSUR

This subroutine generates an N1xN2 point surface (Coon's patch, cf. Ref. 3) by tensor-product interpolation from the four corners. The routine receives the Cartesian coordinates of the four corner points, and two slope vectors, at each at the appropriate positions in the arrays R(3,N1,N2), T1(3,N1,N2), and T2(3,N1,N2), these being equivalenced with CORD, TANGEN1, and TANGEN2, respectively, in COMMON/RAY/.

The surface is generated by bi-cubic interpolation from the four corners:



Here the coordinates u,v vary as ξ^1, ξ^2, but normalized to the range 0-1. The interpolation is given by (cf. Ref. 5),

$$\underset{\sim}{r}(u,v) = B(u)\ S\ B^T(v)$$

where

$$S(4,4,\_) = \begin{bmatrix} \underset{\sim}{r}(0,0) & \underset{\sim}{r}(0,1) & \underset{\sim}{r}_v(0,0) & \underset{\sim}{r}_v(0,1) \\ \underset{\sim}{r}(1,0) & \underset{\sim}{r}(1,1) & \underset{\sim}{r}_v(1,0) & \underset{\sim}{r}_v(1,1) \\ \underset{\sim}{r}_u(0,0) & \underset{\sim}{r}_u(0,1) & 0 & 0 \\ \underset{\sim}{r}_u(1,0) & \underset{\sim}{r}_u(1,1) & 0 & 0 \end{bmatrix}$$

with the first subscript running down the columns, and the second across the rows. (The last subscript indicates the component.) The parentheses refer to the corners as indicated on the figure, and the subscripts indicate partial differentiation. Here the torsion vectors $\underset{\sim}{r}_{uv}$ are set to zero. Also

$$B(\alpha) = \begin{bmatrix} 1 - 3\alpha^2 + 2\alpha^3 \\ 3\alpha^2 - 2\alpha^3 \\ \alpha - 2\alpha^2 + \alpha^3 \\ -\alpha^2 + \alpha^3 \end{bmatrix}$$

The slope vectors received in T1 and T2 are $\underset{\sim}{r}_{\xi^1}$ and $\underset{\sim}{r}_{\xi^2}$, and hence are converted to $\underset{\sim}{r}_u, \underset{\sim}{r}_v$ by

$$\underset{\sim}{r}_u = (N1 - 1)\underset{\sim}{r}_{\xi^1}$$

$$\underset{\sim}{r}_v = (N2 - 1)\underset{\sim}{r}_{\xi^2}$$

The code uses the following notation:

S	:	Q(4,4,3)
u,v	:	X(1,1), X(2,1)

u^2, v^2	:	X(1,2), X(2,2)
u^3, v^3	:	X(1,3), X(2,3)
B(u),B(v)	:	V(4,1), V(4,2)
$S\ B^T(v)$	:	QB(3,4)

36. <u>CORPAR</u>

This subroutine determines the two surface parametric(spline) coordinates of the points on a surface spline that are closest to the points received in the argument array R(3,N1,N2), the dimensions N1,N2 also being received as arguments. The spline is received in SUR(3, NI1,NI2,0:3) in COMMON/SPLINE/. The number of points on the spline, NI1,NI2, are received through equivalence with POINTSI(2) in COMMON/PAR/. The two spline coordinates of each point located on the spline surface are returned in R(2,N1,N2).

With $\underset{\sim}{r}$ the position vector of a general point on the spline surface, and $\underset{\sim}{R}$ that of a certain point in R(3,N1,N2), the point on the spline surface that is closest to $\underset{\sim}{R}$ is obtained by minimizing $|\underset{\sim}{R} - \underset{\sim}{r}|$. This occurs where

$$|\underset{\sim}{R} - \underset{\sim}{r}|_u = |\underset{\sim}{R} - \underset{\sim}{r}|_v = 0$$

with u,v the spline coordinates on the surface.

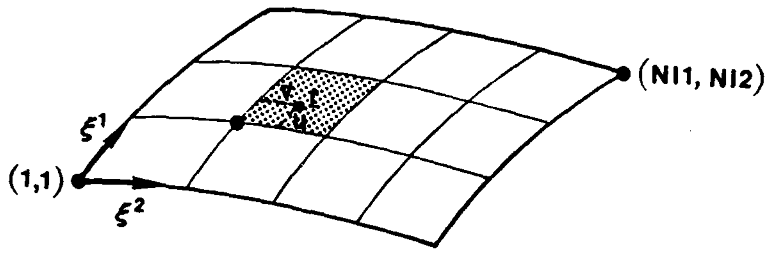
The two equations to be solved for the two spline coordinates u,v are thus

$$(\underset{\sim}{R} - \underset{\sim}{r}) \cdot \underset{\sim}{r}_u = 0$$

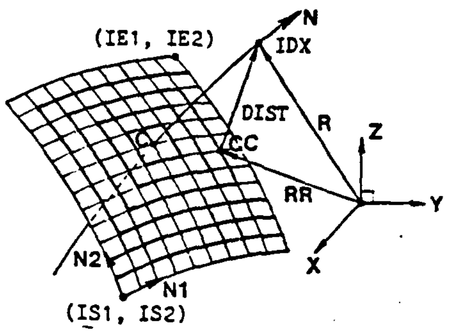
$$(\underset{\sim}{R} - \underset{\sim}{r}) \cdot \underset{\sim}{r}_v = 0$$

This is done as follows.

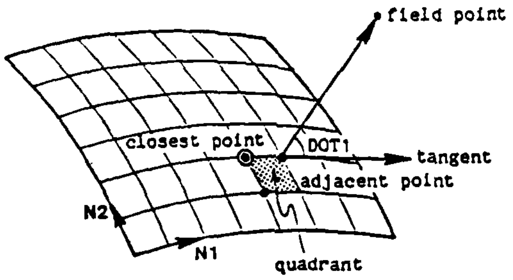
The full spline coordinates of a point on the surface are $\xi^1 + u$, $\xi^2 + v$, where ξ^1 varies from 1 to NI1, and ξ^2 from 1 to NI2, with u,v on the range 0-1:



It is thus first necessary to locate the cell within which the desired closest point $\underline{r}$ lies. To this end, the surface is swept in ever expanding squares centered on a mid-point on the surface to determine the grid point $\underline{r}_0$ on the surface that is closest to the point $\underline{R}$. (These 'squares' are actually limited by the section edges, of course, and hence may become rectangles.):



The desired point then must lie in one of the four cells adjacent to this point. Therefore the dot products of the vector $\underline{R}-\underline{r}_0$ with $\underline{r}_u$ and $\underline{r}_v$ at $\underline{r}_0$ are calculated, and the cell over which $\underline{R}$ lies is determined from the signs of these dot products:



The spline coefficients for the cell indicated are then set in S(4,4,_) as in Appendix F, so that a general position vector on this cell is given by

$$\underline{r}(u,v) = B(u)\ S\ B^T(v)$$

with B given in Appendix F. The equations to be solved for u,v are then

$$F1(u,v) = [R - B(u)\ S\ B^T(v)] \cdot [B'(u)\ S\ B^T(v)] = 0$$

$$F2(u,v) = [R - B(u)\ S\ B^T(v)] \cdot [B(u)\ S\ B^{T'}(v)] = 0$$

These are solved by Newton iteration defined by

$$\frac{dF}{dq} = \Delta q = -F$$

where

$$q = \begin{bmatrix} u \\ v \end{bmatrix}, \qquad F = \begin{bmatrix} F1 \\ F2 \end{bmatrix}$$

and the 2x2 matrix, dF/dq = J(2,2), is given by

$$J(2,2) = \begin{bmatrix} F1_u & F1_v \\ F2_u & F2_v \end{bmatrix}$$

with

$$F1_u = (R - BSB^T) \cdot (B''SB^T) - |B'SB^T|^2$$

$$F1_v = F2_u = (R - BSB^T) \cdot (B'SB^{T'}) - (BSB^{T'}) \cdot (B'SB^T)$$

$$2_v = (R - BSB^T) \cdot (BSB^{T''}) - |BSB^{T'}|^2$$

The cod[illegible] uses t[illegible] following notation:

S	:	Q(4, .3)
u,v	:	X1,X2
B(u), B(v)	:	G1(4), G2(4)
B'(u), B'(v)	:	GP1(4), GP2(4)

$B''(u)$, $B''(v)$	:	GPP1(4), GPP2(4)
$Q\ B^T(v)$	:	QG2(3,4)
$Q\ B^{T'}(v)$	:	QGP2(3,4)
$Q\ B^{T''}(v)$	:	QGPP2(3,4)
$\underset{\sim}{r}(u,v)$	:	V1(3)
$B'\ S\ B^T$	:	V2(3)
$B\ S\ B^{T'}$	:	V3(3)
$B''\ S\ B^T$	:	V4(3)
$B'\ S\ B^T$	:	V5(3)
$B\ S\ B^{T''}$	:	V6(3)
F1,F2	:	F1,F2
$F1_u, F1_v$	:	F11,F12
$F2_u, F2_v$	:	F21,F22

$(\underset{\sim}{R} - \underset{\sim}{r}_0) \cdot \underset{\sim}{r}_u$, $(\underset{\sim}{R} - \underset{\sim}{r}_0) \cdot \underset{\sim}{r}_v$: DOT1,DOT2

In addition, the grid coordinates of the point $\underset{\sim}{r}_0$ are C1,C2, and the signs of the dot products are L1,L2.

37. TANPAR

This subroutine generates the derivatives of the spline coordinates from slope vectors received in the argument array T(3,N1,N2) at the N1xN2 set of points defined by the surface parametric(spline) coordinates received in the argument array R(2,N1,N2). The values N1 and N2 are also received as arguments. The surface spline is received in the array SUR(3,NI1,NI2,0:3) in COMMON/SPLINE/.

The grid coordinates of the cell on the surface within which the point lies are first determined as the integer parts of the two coordinates received in R. The spline coordinates u,v (which vary on the range 0-1) are then the difference between the values in R and these integer values. Since the Cartesian coordinates of a general point on the spline surface are given by (Appendix F)

$$\underset{\sim}{r} = B(u)\ S\ B^T(v)$$

we have

$$\underset{\sim}{r}_u = B'\ S\ B^T$$

$$\underset{\sim}{r}_v = B\ S\ B^{T'}$$

The slope vectors received in T are $\underset{\sim}{r}_\xi$, which are related to $\underset{\sim}{r}_u$ and $\underset{\sim}{r}_v$ by

$$\underset{\sim}{r}_\xi = \underset{\sim}{r}_u u_\xi + \underset{\sim}{r}_v v_\xi$$

Then

$$\begin{bmatrix} |\underset{\sim}{r}_u|^2 & \underset{\sim}{r}_u \cdot \underset{\sim}{r}_v \\ \underset{\sim}{r}_u \cdot \underset{\sim}{r}_v & |\underset{\sim}{r}_v|^2 \end{bmatrix} \begin{bmatrix} u_\xi \\ v_\xi \end{bmatrix} = \begin{bmatrix} \underset{\sim}{r}_u \cdot \underset{\sim}{r}_\xi \\ \underset{\sim}{r}_v \cdot \underset{\sim}{r}_\xi \end{bmatrix}$$

determines u_ξ and v_ξ. These derivatives are returned in T(2,N1,N2).

The code uses the following notation:

u,v	:	X1,X2
S	:	Q(4,4,3)
B(u) and B'(u)	:	V(4,1)

B(v) and B'(v)	:	V(4,2)
$S\,B^T$ and $S\,B^{T'}$	:	QB(3,4)
$\underset{\sim}{r}_u, \underset{\sim}{r}_v$	:	T1(3),T2(3)
$\lvert\underset{\sim}{r}_u\rvert^2, \lvert\underset{\sim}{r}_v\rvert^2, \underset{\sim}{r}_u \cdot \underset{\sim}{r}_v$	:	D11,D22,D12
$\underset{\sim}{r}_u \cdot \underset{\sim}{r}_\xi, \underset{\sim}{r}_v \cdot \underset{\sim}{r}_\xi$	:	DOT1,DOT2

38. PARCOR

This subroutine generates the Cartesian coordinates from surface parametric(spline) coordinates by bi-cubic interpolation on the spline. The N1xN2 lattice of parametric coordinates is received in the argument array R(2,N1,N2), and its dimensions are received as the arguments N1,N2. The surface spline is received in the array SUR(3,NI1,NI2,0:3) in COMMON/SPLINE/.

The grid coordinates of the cell on the surface within which the point lies are first determined as the integer parts of the two coordinates received in R. The spline coordinates u,v (which vary on the range 0-1) are then the difference between the values in R and these integer values. The Cartesian coordinates of the point on the spline surface corresponding to the spline coordinates are then given by (Appendix F)

$$\underset{\sim}{r} = B(u)\ S\ B^T(v)$$

The code uses the following notation:

u,v	:	X1,X2
S	:	Q(4,4,3)

B(u), B(v)	:	V(4,1), V(4,2)
S B^T(v)	:	QB(3,4)

39. SPLNSUR

This subroutine receives a surface(curve) in the argument array R(3,N1,N2), and its dimensions N1,N2 also as arguments, and returns the surface spline in the array SUR(3,N1,N2,0:3) in COMMON/SPLINE/.

The spline array SUR(3,N1,N2,_) contains the Cartesian coordinates of the points on the surface, with a 0 for the last subscript, and the derivatives $\mathbf{r}_u$, $\mathbf{r}_v$, $\mathbf{r}_{uv}$ with 1,2,3 for the last subscript. The routine transfers the Cartesian coordinates from R into SUR(3,N1,N2,0) and then calls SURSPL to calculate $\mathbf{r}_u$, $\mathbf{r}_v$, and $\mathbf{r}_{uv}$.

40. SURSPL

This subroutine splines an N1xN2 surface as described in Appendix G. The surface dimensions are received as arguments. The surface is received in the array SUR(3,N1,N2,0). The routine transfers each curve into CUR(3,N1 or N2,0). The routine calls CURSPL to spline each curve on the surface, with the argument "QUAD" for quadratic ends when calculating $\mathbf{r}_u$ and $\mathbf{r}_v$, and with "SPECIF" for specified slopes at the ends when calculating $\mathbf{r}_{uv}$. These derivatives are transferred from CUR(3,N1 or N2,1) into SUR after each return from CURSPL.

41. CURSPL

This subroutine splines an NP-point curve as described in Appendix G. The number of points, NP, on the curve is received as an argument, and the curve is received as an argument, and the curve is received in the array CUR(3,NP,0). The spline ends will have zero curvature(natural spline) if the alphanumeric integer argument TYPE is equal to "NATURAL", constant curvature for "QUAD", or specified slope for "SPECIF". In the latter case the specified slope is received in CUR(3,1,1) and CUR(3,NP,1). The derivative vectors are returned in CUR(3,NP,1).

The code uses the following notation:

CUR(3,NP,0)	:	$\underset{\sim}{r}$
CUR(3,NP,1)	:	$\underset{\sim}{r}_\xi$
F	:	right hand side of Eq. (E-2)
AA	:	coefficient of $\underset{\sim}{s}_2$ and $\underset{\sim}{s}_{N-1}$ for i=2 and NP-1
BB	:	coefficient of $\underset{\sim}{s}_3$ and $\underset{\sim}{s}_{N-2}$ for i=2 and NP-1

APPENDIX A

DISTRIBUTION FUNCTIONS

In general, interpolation between $\underset{\sim}{r}_1$ at $\xi=0$ and $\underset{\sim}{r}_2$ at $\xi=I$ can be written

$$\underset{\sim}{r}(\xi) = \phi(\frac{\xi}{I})\, \underset{\sim}{r}_2 + [1 - \phi(\frac{\xi}{I})]\, \underset{\sim}{r}_1 \tag{1}$$

where ϕ can be any function such that $\phi(0)=0$ and $\phi(1)=1$. Here we have taken $\phi_1=1-\phi$ and $\phi_2=\phi$. The linear polynomial case is obtained here with $\phi(\frac{\xi}{I}) = \frac{\xi}{I}$. The function ϕ in this form may contain parameters which can be determined so as to match the slope at the boundary, or to match interior points and slopes. (The ξ in this appendix is one less than the point index on the curve in the code, i.e., the points are numbered from 1 to N as ξ varies from 0 to I, so that $I=N-1$.)

The interpolation function, ϕ, in this form is often referred to as a "stretching" function, and the most widely used function has been the exponential:

$$\phi(\frac{\xi}{I}) = \frac{\exp(\frac{\alpha\xi}{I}) - 1}{\exp(\alpha) - 1} \tag{2}$$

where α is a parameter that can be determined to match the slope at a boundary. Thus, since, from Eq. (1)

$$\underset{\sim}{r}_\xi = \frac{1}{I} (\underset{\sim}{r}_2 - \underset{\sim}{r}_1)\, \phi' \tag{3}$$

we can determine α from the equation

$$(\underset{\sim}{r}_\xi)_1 = \frac{\underset{\sim}{r}_2 - \underset{\sim}{r}_1}{I} \frac{\alpha}{\exp(\alpha) - 1} \tag{4}$$

with $(\underset{\sim}{r}_\xi)_1$ specified.

The truncation error is strongly affected by the point distribution, and studies of distribution functions have been made in that regard. The exponential, while reasonable, is not the best choice when the variation of spacing is large, and polynomials are not suitable in this case. The better choices are the hyperbolic tangent and the hyperbolic sine. The hyperbolic sine gives a more uniform distribution in the immediate vicinity of the minimum spacing, and thus has less error in this region, but the hyperbolic tangent has the better overall distribution (cf. Ref. 4,5). These functions are implemented as follows (following Ref. 4), with the spacing specified at either or both ends, or a point in the interior, of a point distribution on a curve.

Let arc length, s, vary from 0 to 1 as ξ varies from 0 to I: $s(0)=0$, $s(I)=1$. Then let the spacing be specified at $\xi=0$ and $\xi=I$:

$$s_\xi(0) = \Delta s_1, \qquad s_\xi(I) = \Delta s_2 \tag{5}$$

The hyperbolic tangent distribution is then constructed as follows.

First,

$$A = \frac{\sqrt{\Delta s_2}}{\sqrt{\Delta s_1}} \tag{6}$$

$$B = I\sqrt{\Delta s_1 \Delta s_2} \tag{7}$$

Then the following nonlinear equation is solved for δ:

$$\frac{\sinh\delta}{\delta} = \frac{1}{B} \tag{8}$$

The arc length distribution then is given by

$$s(\xi) = \frac{u(\xi)}{A + (1-A)u(\xi)} \tag{9}$$

where

$$u(\xi) = \frac{1}{2}\{1 + \frac{\tanh[\delta(\frac{\xi}{I} - \frac{1}{2})]}{\tanh(\frac{\delta}{2})}\} \tag{10}$$

If this is applied to a straight line on which $\underset{\sim}{r}$ varies from $\underset{\sim}{r}_0$ to $\underset{\sim}{r}_I$ we have for the point locations:

$$\underset{\sim}{r}(\xi) = \underset{\sim}{r}_0 + (\underset{\sim}{r}_I - \underset{\sim}{r}_0)s(\xi) \tag{11}$$

The points are then located by taking integer values of ξ:

$$\xi = 0,1,2\ldots,I$$

Clearly the arc length distribution, $s(\xi)$, here is the function ϕ of Eq. (1).

Note that B is essentially the ratio of the specified spacing to the linear spacing, 1/I. If B is greater than unity, i.e. if the specified spacing exceeds the linear spacing, the hyperbolic functions all revert to circular functions in all the relations of this appendix.

With the spacing Δs specified at only $\xi=0$, the construction proceeds as follows. First B is calculated from

$$B = I\Delta s \tag{12}$$

and Eq. (8) is solved for δ. The arc length distribution then is given by

$$s(\xi) = 1 + \frac{\tanh[\frac{\delta}{2}(\frac{\xi}{I} - 1)]}{\tanh(\frac{\delta}{2})} \tag{13}$$

With the spacing specified only at $\xi = I$ the procedure is the same, except that Eq. (13) is replaced by

$$s(\xi) = \frac{\tanh\left(\frac{\delta}{2}\frac{\xi}{I}\right)}{\tanh\left(\frac{\delta}{2}\right)} \tag{14}$$

If the spacing Δs is specified at only an interior point $s = \sigma$, B is again calculated from Eq. (12), and then δ is determined as the solution of

$$1 + \left(\frac{B}{\sigma\delta}\right)^2 = \left(\frac{\cosh\delta - 1 + \frac{1}{\sigma}}{\sinh\delta}\right)^2 \tag{15}$$

The value of ξ at which $s = \sigma$ is obtained by solving the nonlinear equation

$$\chi = \frac{I}{\delta}\tanh^{-1}\left(\frac{\sinh\delta}{\frac{1}{\sigma} + \cosh\delta - 1}\right) \tag{16}$$

The arc length distribution then is given by

$$s(\xi) = \sigma\left\{\frac{1 + \sinh[\delta(\frac{\xi - \chi}{I}]}{\sinh(\delta\frac{\chi}{I})}\right\} \tag{17}$$

This last distribution is based on the hyperbolic sine. From this, a distribution based on the hyperbolic sine with the spacing specified at one end can be derived. Here B is evaluated from Eq. (12), and then δ is determined as the solution of

$$\frac{\sinh\delta}{\delta} = \frac{1}{B} \tag{18}$$

The arc length distribution then is given by

$$s(\xi) = \frac{\sinh(\delta\frac{\xi}{I})}{\sinh \delta} \tag{19}$$

if the spacing is specified at $\xi=0$. With the specification at $\xi=I$, the distribution is

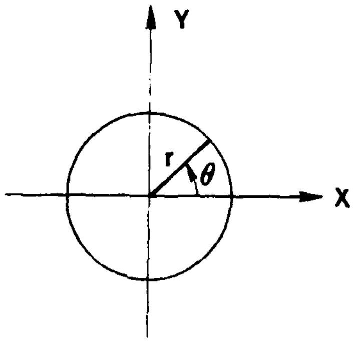
$$s(\xi) = 1 - \frac{\sinh[\delta(1 - \frac{\xi}{I})]}{\sinh \delta} \tag{20}$$

APPENDIX B

CONIC SECTION CURVES

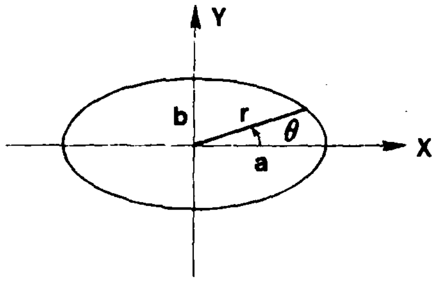
The equations for the conic-section curves in the x-y plane are developed as follows. The code notation corresponds closely to that used here.

1. Circle



$r(\theta)$ = constant

2. Ellipse



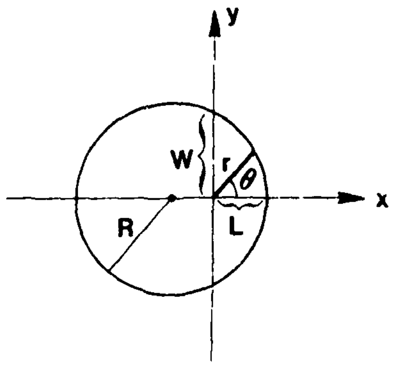
The equation of an ellipse is

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

Substitution of $x = r\cos\theta$ and $y = r\sin\theta$ *yields*

$$r(\theta) = \left[\frac{\cos^2\theta}{a^2} + \frac{\sin^2\theta}{b^2}\right]^{-1/2}$$

3. <u>Circular arc</u>



The equation of a circle centered on the x-axis is

$$(x - h)^2 + y^2 = A^2$$

Evaluation at the intersections on the positive axes yields

$$(L - h)^2 = A^2$$

$$h^2 + W^2 = A^2$$

so that

$$A = \frac{1}{2}\left(L + \frac{W^2}{L}\right)$$

$$h = L - A$$

Substitution for x and y in terms of r and θ then yields a quadratic equation for r, the solution of which is

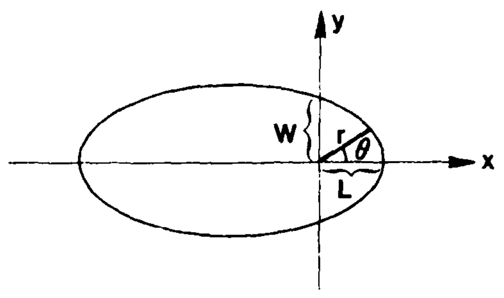
$$r(\theta) = \frac{-b + \sqrt{b^2 - 4c}}{2}$$

with

$$b = -2h \cos\theta$$

$$c = h^2 - A^2$$

4. Elliptical arc



The equation for an ellipse with its major axis on the x-axis is, with e the eccentricity,

$$(x - h)^2 + \frac{y^2}{1-e^2} = A^2$$

Evaluation at the intersections on the positive axes yields

$$(L - h)^2 = A^2$$

$$h^2 + \frac{W^2}{1-e^2} = A^2$$

from which

$$h = \frac{1}{2}\left[L - \frac{W^2}{L(1-e^2)}\right]$$

$$A = \sqrt{h^2 + \frac{W^2}{1-e^2}}$$

Substitution for x and y yields a quadratic equation for r with the solution

$$r(\theta) = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

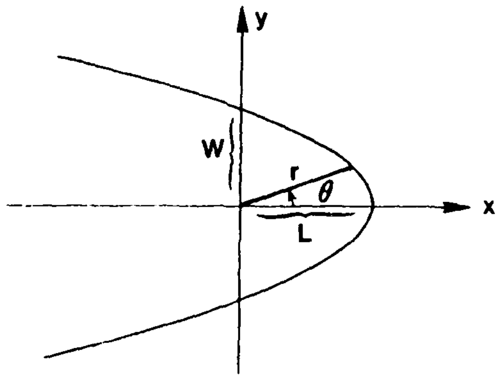
where

$$a = \cos^2\theta + \frac{\sin^2\theta}{1-e^2}$$

$$b = -2h \cos\theta$$

$$c = h^2 - A^2$$

5. <u>Parabola</u>



The equation of a parabola astride the x-axis is

$$\alpha(x - L) + y^2 = 0$$

Evaluation at the intersection on the positive y-axis yields

$$\alpha = \frac{W^2}{L}$$

Substitution for x and y then produces a quadratic equation for r from which

$$r(\theta) = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

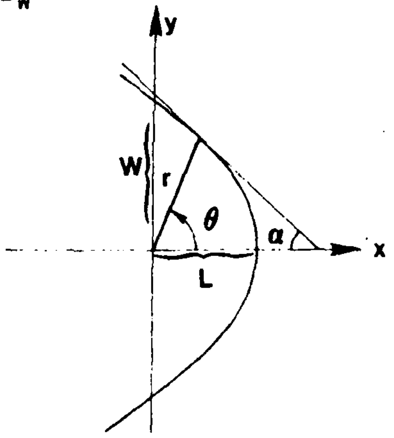
with

$$a = \sin^2\theta$$

$$b = \alpha \cos\theta$$

$$c = -W^2$$

6. Hyperbola



The equation for a hyperbola astride the x-axis is

$$(x - h)^2 - \frac{y^2}{\tan^2\alpha} = A^2$$

Evaluation at the intersections on the positive axes yields

$$(L - h)^2 = A^2$$

$$h^2 - \frac{W^2}{\tan^2\alpha} = A^2$$

which yield

$$h = \frac{1}{2}\left(L + \frac{W^2}{L\tan^2\alpha}\right)$$

$$A = \sqrt{h^2 - \frac{W^2}{\tan^2\alpha}}$$

Substitution for x and y then produces a quadratic equation for r with the solution

$$r(\theta) = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

with

$$a = \cos^2\theta - \frac{\sin^2\theta}{\tan^2\alpha}$$

$$b = -2h\cos\theta$$

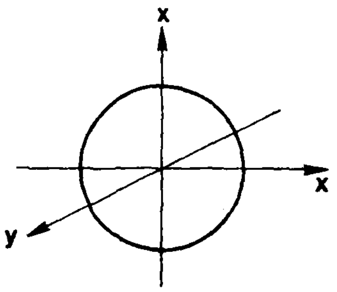
$$c = h^2 - A^2$$

APPENDIX C

CONIC SECTION SURFACES

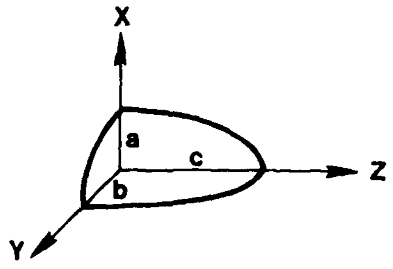
The equations for the conic-section surfaces are developed as follows: (The code notation follows closely that used here.)

1. Sphere



$r(\theta,\psi) = \text{constant}$

2. Ellipsoid



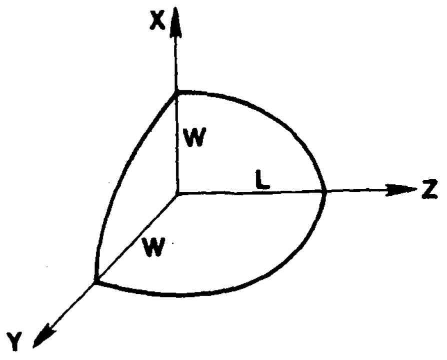
The equation for an ellipsoid is

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$$

Substitution of $x = r \sin\theta \cos\psi$, $y = \sin\theta \sin\psi$, and $z = r \cos\theta$, where θ and ψ are analogous to latitude and longitude respectively, (with the z-axis as the polar axis) yields

$$r(\theta,\psi) = \left[\frac{\sin^2\theta \cos^2\psi}{a^2} + \frac{\sin^2\theta \sin^2\psi}{b^2} + \frac{\cos^2\theta}{c^2}\right]^{-1/2}$$

3. <u>Spherical segment</u>



The equation of a sphere centered on the z-axis is

$$x^2 + y^2 + (z - h)^2 = A^2$$

Evaluation at the intersections on the positive axes gives

$$(L - h)^2 = A^2$$

$$W^2 + h^2 = A^2$$

from which

$$A = \frac{1}{2}\left(L + \frac{W^2}{L}\right)$$

$$h = L - R$$

Substitution of x, y, and z in terms of θ and ψ then yields a quadratic equation for r, the solution of which is

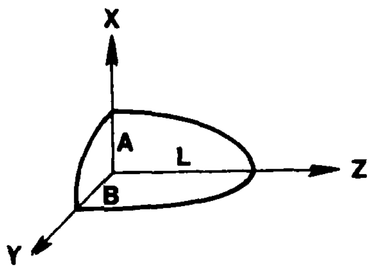
$$r(\theta,\psi) = \frac{-b + \sqrt{b^2 - 4c}}{2}$$

with

$$b = -2h \cos\theta$$

$$c = h^2 - A^2$$

4. Ellipsoidal segment



The equation for an ellipsoid with its major axis on the z-axis is, with e the eccentricity in the x-z plane,

$$\frac{x^2}{\alpha^2} + \frac{y^2}{\beta^2} + \frac{(z - h)^2}{\gamma^2} = 1$$

Evaluation at the intersections on the positive axes yields

$$(L - h)^2 = \gamma^2$$

$$\frac{A^2}{\alpha^2} + \frac{h^2}{\gamma^2} = 1$$

$$\frac{B^2}{\beta^2} + \frac{h^2}{\gamma^2} = 1$$

from which

$$h = \frac{1}{2}\left[L - \frac{A^2}{L(1-e^2)}\right]$$

$$\gamma = \sqrt{h^2 + \frac{A^2}{1-e^2}}$$

$$\alpha = \frac{A}{\sqrt{1 - \frac{h^2}{\gamma^2}}}$$

$$\beta = \frac{B}{\sqrt{1 - \frac{h^2}{\gamma^2}}}$$

where e is the eccentricity in the x-z plane: $\alpha = \gamma\sqrt{1-e^2}$. Substitution for x, y, and z yields a quadratic equation for r with the solution

$$r(\theta,\psi) = \frac{- b + \sqrt{b^2 - 4ac}}{2a}$$

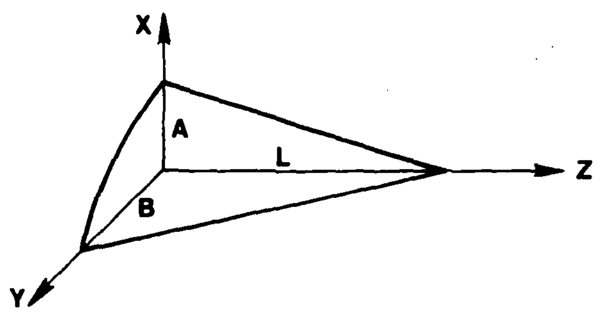
with

$$a = \left(\frac{\cos^2\psi}{\alpha^2} + \frac{\sin^2\psi}{b^2}\right)\sin^2\theta + \frac{\cos^2\theta}{\gamma^2}$$

$$b = -\frac{2h\cos\theta}{\gamma^2}$$

$$c = \frac{h^2}{\gamma^2} - 1$$

5. <u>Elliptic Cone</u>



The equation for an elliptic cone with its axis on the z-axis is

$$\frac{x^2}{\alpha^2} + \frac{y^2}{\beta^2} - (z - h)^2 = 0$$

Evaluation at the intersections on the positive axes yields

$$(L - h)^2 = 0$$

$$\frac{A^2}{\alpha^2} + L^2 = 0$$

$$\frac{B^2}{\beta^2} + L^2 = 0$$

so that

$$h = L$$

$$\alpha = \frac{A}{L}$$

$$\beta = \frac{B}{L}$$

Substitution for x,y, and z then produces a quadratic equation for r, the solution of which is

$$r(\theta,\psi) = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

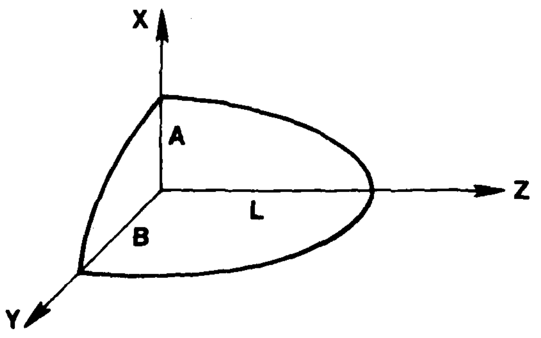
with

$$a = \left(\frac{\cos^2\psi}{\alpha^2} + \frac{\sin^2\psi}{\beta^2}\right)\sin^2\theta - \cos^2\theta$$

$$b = 2h\cos\theta$$

$$c = -h^2$$

6. <u>Elliptic paraboloid</u>



The equation of an elliptic paraboloid astride the z-axis is

$$\frac{x^2}{\alpha^2} + \frac{y^2}{\beta^2} + (z - h) = 0$$

Evaluation at the intersections on the positive axes yields

$$L - h = 0$$

$$\frac{A^2}{\alpha^2} - L = 0$$

$$\frac{B^2}{\beta^2} - L = 0$$

which gives

$$h = L$$

$$\alpha = \frac{A}{\sqrt{L}}$$

$$\beta = \frac{B}{\sqrt{L}}$$

Substitution for x,y, and z gives a quadratic equation for r which produces

$$r(\theta,\psi) = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

with

$$a = \left(\frac{\cos^2\psi}{\alpha^2} + \frac{\sin^2\psi}{\beta^2}\right)\sin^2\theta$$

$$b = \cos\theta$$

$$c = -L$$

APPENDIX D

CUBIC SPACE CURVE

The curve is generated as a cubic polynomial:

$$\underset{\sim}{r} = \underset{\sim}{r}_0 + \underset{\sim}{r}_0'\xi + \underset{\sim}{a}\xi^2 + \underset{\sim}{b}\xi^3$$

where the subscript indicates values at the first end, where $\xi=0$. Evaluation at the other end ($\xi=1$) yields

$$\underset{\sim}{r}_1 = \underset{\sim}{r}_0 + \underset{\sim}{r}_0' + \underset{\sim}{a} + \underset{\sim}{b}$$

$$\underset{\sim}{r}_1' = \underset{\sim}{r}_0' + 2\underset{\sim}{a} + 3\underset{\sim}{b}$$

Then

$$\underset{\sim}{a} = 3(\underset{\sim}{r}_1 - \underset{\sim}{r}_0) - 2\underset{\sim}{r}_0' - \underset{\sim}{r}_1'$$

$$\underset{\sim}{b} = \underset{\sim}{r}_0' + \underset{\sim}{r}_1' - 2(\underset{\sim}{r}_1 - \underset{\sim}{r}_0)$$

APPENDIX E

RADIUS OF CURVATURE

The unit tangent, $\underset{\sim}{T}$, to a curve is given by the derivative of the position vector with respect to arc length, s:

$$\underset{\sim}{T} = \frac{\underset{\sim}{r}_s}{|\underset{\sim}{r}_s|}$$

The principal normal, $\underset{\sim}{N}$, then is given by

$$\kappa \underset{\sim}{N} = \underset{\sim}{T}_s$$

where κ is the curvature. Now

$$\underset{\sim}{T}_s = \frac{|\underset{\sim}{r}_s|^2 \underset{\sim}{r}_{ss} - \underset{\sim}{r}_s(\underset{\sim}{r}_s \cdot \underset{\sim}{r}_{ss})}{|\underset{\sim}{r}_s|^3}$$

and then, by an identity,

$$|\underset{\sim}{T}_s| = \frac{|\underset{\sim}{r}_s \times \underset{\sim}{r}_{ss}|}{|\underset{\sim}{r}_s|^2}$$

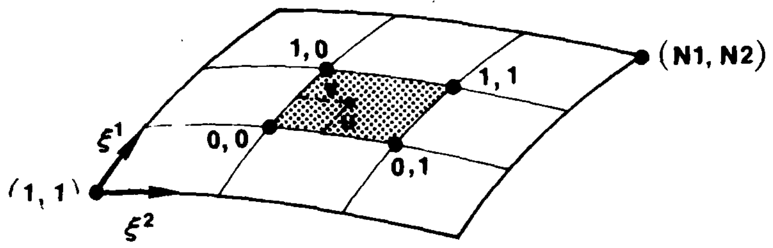
so that the radius of curvature is given by

$$\frac{1}{\rho} = \kappa = \frac{|\underset{\sim}{r}_s \times \underset{\sim}{r}_{ss}|}{|\underset{\sim}{r}_s|^2}$$

APPENDIX F

SURFACE(CURVE) SPLINE

The spline coordinates of a point on an N1xN2 surface(curve) are ξ^1+u and ξ^2+v, where ξ^1, ξ^2 are the integer indices of a grid point varying as $\xi^1=1,2,\text{--},N1$ and $\xi^2=1,2,\text{--},N2$. Thus u,v vary on the range 0-1:



The Cartesian coordinates of a general point on the cell forward of ξ^1, ξ^2 are given by

$$\underset{\sim}{r}(u,v) = B(u)\ S\ B^T(v)$$

where

$$S(4,4,\_) = \begin{bmatrix} \underset{\sim}{r}(0,0) & \underset{\sim}{r}(0,1) & \underset{\sim}{r}_v(0,0) & \underset{\sim}{r}_v(0,1) \\ \underset{\sim}{r}(1,0) & \underset{\sim}{r}(1,1) & \underset{\sim}{r}_v(1,0) & \underset{\sim}{r}_v(1,1) \\ \underset{\sim}{r}_u(0,0) & \underset{\sim}{r}_u(0,1) & \underset{\sim}{r}_{uv}(0,0) & \underset{\sim}{r}_{uv}(0,1) \\ \underset{\sim}{r}_u(1,0) & \underset{\sim}{r}_u(1,1) & \underset{\sim}{r}_{uv}(0,1) & \underset{\sim}{r}_{uv}(1,1) \end{bmatrix}$$

with the first subscript running down the columns, and the second across the rows. (The last subscript indicates the Cartesian component.) The parantheses refer to the corners indicated on the figure above, and the subscripts indicate partial differentiation. Also

$$B(\alpha) = \begin{bmatrix} 1 - 3\alpha^2 + 2\alpha^3 \\ 3\alpha^2 - 2\alpha^3 \\ \alpha - 2\alpha^2 + \alpha^3 \\ -\alpha^2 + \alpha^3 \end{bmatrix}$$

For a curve, S contains only the top row and

$$\underline{r}(u) = S\ B(u)$$

APPENDIX G

SPLINE

The set of grid points $\underset{\sim}{r}(\xi)$, $\xi=1,2,\text{--},N$, is splined with the curvilinear coordinate ξ as the parameter. Thus on the interval $\xi_i \leq \xi \leq \xi_i+1$, the curve is given by the cubic polynomial

$$\underset{\sim}{r}(u) = \frac{1}{6}\,\underset{\sim}{s}_i(1-u)^3 + \frac{1}{6}\,\underset{\sim}{s}_{i+1}u^3$$

$$+ (\underset{\sim}{r}_i - \frac{1}{6}\,\underset{\sim}{s}_i)(1-u) + (\underset{\sim}{r}_{i+1} - \frac{1}{6}\,\underset{\sim}{s}_{i+1})u \qquad (1)$$

where $\underset{\sim}{s} = \underset{\sim}{r}_{\xi\xi}$ and $0 \leq u \leq 1$, i.e., $\xi = \xi_i + u$, and $\underset{\sim}{r}_i = \underset{\sim}{r}(\xi_i)$, $\underset{\sim}{r}_{i+1} = r(\xi_{i+1})$, etc.

The second derivatives are determined by the tridiagonal solution of

$$\underset{\sim}{s}_{i-1} + 4\underset{\sim}{s}_i + \underset{\sim}{s}_{i+1} = 6(\underset{\sim}{r}_{i+1} - 2\underset{\sim}{r}_i + \underset{\sim}{r}_{i-1}) \qquad (2)$$

for $i=2,3,\text{--},N-1$. The end values are determined in one of three ways:

(1) zero curvature (natural spline):

$$\underset{\sim}{s}_1 = \underset{\sim}{s}_N = 0$$

(2) constant curvature (quadratic ends):

$$\underset{\sim}{s}_1 = 2\underset{\sim}{s}_2 - \underset{\sim}{s}_3$$

$$\underset{\sim}{s}_N = 2\underset{\sim}{s}_{N-1} - \underset{\sim}{s}_{N-2}$$

(3) specified slopes:

$$\underset{\sim}{s}_1 = 3(\underset{\sim}{r}_2 - \underset{\sim}{r}_1) - 3g_1 - \frac{1}{2}\underset{\sim}{s}_2$$

$$\underset{\sim}{s}_N = -3(\underset{\sim}{r}_N - \underset{\sim}{r}_{N-1}) + 3g_N - \frac{1}{2}\underset{\sim}{s}_{N-1}$$

where $g = \underset{\sim}{r}_\xi$.

In the first case above, Eq. (2) is replaced by the following equations for $i=2$ and $N-1$:

$$4\underset{\sim}{s}_2 + \underset{\sim}{s}_3 = 6(\underset{\sim}{r}_3 - 2\underset{\sim}{r}_2 + \underset{\sim}{r}_1)$$

$$4\underset{\sim}{s}_{N-1} + \underset{\sim}{s}_{N-2} = 6(\underset{\sim}{r}_N - 2\underset{\sim}{r}_{N-1} + \underset{\sim}{r}_{N-2})$$

while in the second case

$$6\underset{\sim}{s}_2 = 6(\underset{\sim}{r}_3 - 2\underset{\sim}{r}_2 + \underset{\sim}{r}_1)$$

$$6\underset{\sim}{s}_{N-1} = 6(\underset{\sim}{r}_N - 2\underset{\sim}{r}_{N-1} + \underset{\sim}{r}_{N-2})$$

and in the third case

$$\frac{7}{2}\underset{\sim}{s}_2 + \underset{\sim}{s}_3 = 6(\underset{\sim}{r}_3 - 2\underset{\sim}{r}_2 + \underset{\sim}{r}_1) + 3g_1 - 3(\underset{\sim}{r}_2 - \underset{\sim}{r}_1)$$

$$\frac{7}{2}\underset{\sim}{s}_{N-1} + \underset{\sim}{s}_{N-2} = 6(\underset{\sim}{r}_N - 2\underset{\sim}{r}_{N-1} + \underset{\sim}{r}_{N-2}) - 3g_{N-1} + 3(\underset{\sim}{r}_N - \underset{\sim}{r}_{N-1} + \underset{\sim}{r}_{N-2})$$

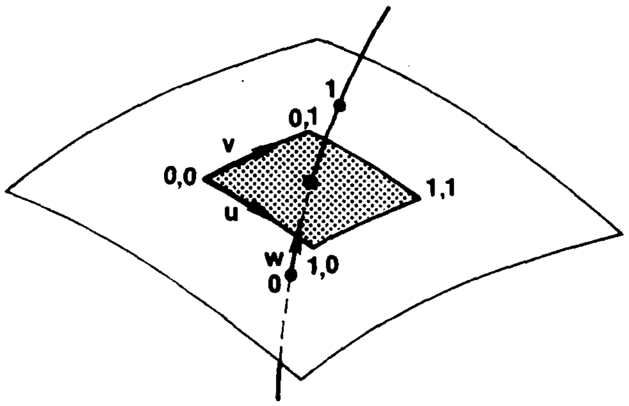
The slopes, $g = \underset{\sim}{r}_\xi$, are given by

$$g_i = \frac{1}{2}(\underset{\sim}{r}_{i+1} - \underset{\sim}{r}_{i-1}) - \frac{1}{12}(\underset{\sim}{s}_{i+1} - \underset{\sim}{s}_{i-1})$$

APPENDIX H

NEWTON ITERATION FOR INTERSECTION

The intersection of a curve spline with a surface spline is determined as follows.



The spline coordinate on a given interval on the curve is w, and those for a given interval on the surface are u,v. All of these vary on the range 0-1. The spline coefficients for the interval on the surface are set in S(4,4,3) as (cf. Ref. 3)

$$S(4,4,\_) = \begin{bmatrix} \underset{\sim}{r}(0,0) & \underset{\sim}{r}(0,1) & \underset{\sim}{r}_v(0,0) & \underset{\sim}{r}_v(0,1) \\ \underset{\sim}{r}(1,0) & \underset{\sim}{r}(1,1) & \underset{\sim}{r}_v(1,0) & \underset{\sim}{r}_v(1,1) \\ \underset{\sim}{r}_u(0,0) & \underset{\sim}{r}_u(0,1) & \underset{\sim}{r}_{uv}(0,0) & \underset{\sim}{r}_{uv}(0,1) \\ \underset{\sim}{r}_u(1,0) & \underset{\sim}{r}_u(1,1) & \underset{\sim}{r}_{uv}(1,0) & \underset{\sim}{r}_{uv}(1,1) \end{bmatrix}$$

with the first subscript running down the columns, and the second across the rows. (The last subscript indicates the Cartesian component.) The parentheses refer to the corners as indicated on the figure above, and the subscripts indicate partial differentiation. Similarly, the spline coefficients for the interval on the curve are set in C(4,3) as

$$C(4,\_) = (\underset{\sim}{r}(0) \quad \underset{\sim}{r}(1) \quad \underset{\sim}{r}_w(0) \quad \underset{\sim}{r}_w(1))$$

a point on the surface spline is then given by

$$\underset{\sim}{r}(u,v) = B(u)\ S\ B^T(v)$$

where

$$B(\alpha) = \begin{bmatrix} 1 - 3\alpha^2 + 2\alpha^3 \\ 3\alpha^2 - 2\alpha^3 \\ \alpha - 2\alpha^2 + \alpha^3 \\ -\alpha^2 + \alpha^3 \end{bmatrix}$$

(In subroutine INTSEC, the notation Q,QC is used for S,C; and the notation X1,X2,XC is used for u,v,w; and G1(4), G2(4), GC(4) is used for B(u), B(v), B(w).) Similarly, a point on the curve spline is given by

$$\underset{\sim}{r}(w) = C\ B(w)$$

The three values u,v,w of the intersection point are determined as the solution of $\underset{\sim}{r}(w) = \underset{\sim}{r}(u,v)$, or

$$F(u,v,w) = B(u)\ S\ B^T(v) - C\ B(w) = 0$$

The Newton iteration then is defined by

$$\frac{dF}{dq}\,\Delta q = -F$$

where

$$q = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

and the 3x3 matrix $\frac{dF}{dq} = J(3,3)$ is given by

$$J(\_,3) = \begin{bmatrix} F_u \\ F_v \\ F_w \end{bmatrix} = \begin{bmatrix} B'(u)\ S\ B^T(v) \\ B(u)\ S\ B^{T'}(v) \\ -C\ B'(w) \end{bmatrix}$$

In subroutine INTSEC the following notation is used:

S,C	: Q(4,4,3), QC(4,3)
u,v,w	: X1,X2,XC
B(u), B(v), B(w)	: G1(4), G2(4), GC(4)
$\underset{\sim}{r}(u,v)$	: RS(3)
$\underset{\sim}{r}(w)$	: RC(3)
F	: F(3)
J	: DF(3,3)
F_u, F_v, F_w	: R1(3), R2(3), R3(3)
B'(u), B'(v), B'(w)	: GP1(4), GP2(4), GPC(4)
$Q\ B^T(v)$	: QG2(3,4)
$Q\ B^{T'}(v)$	: QGP2(3,4)

REFERENCES

1. GORDON, WILLIAM J. and THIEL, LINDA C., "Transfinite Mappings and Their Application to Grid Generation", Numerical Grid Generation, Joe F. Thompson, (Ed.), North-Holland, 1982.

2. THOMPSON, J.F., WARSI, Z.U.A., and MASTIN, C. W., Numerical Grid Generation: Foundations and Applications, North-Holland, 1985.

3. FAUX, I.D. and PRATT, M.J., Computational Geometry for Design and Manufacture, Ellis Horwood, 1979.

4. VINOKUR, MARCEL, "On One-Dimensional Stretching Functions for Finite-Difference Calculations", Journal of Computational Physics, 50, 215, 1983.

5. THOMPSON, J.F., and MASTIN, C. WAYNE, "Order of Difference Expressions on Curvilinear Coordinate Systems", Advances in Grid Generation, FED-Vol. 5, Ed. K.N. Ghia and U. Ghia, ASME Applied Mechanics, Bioengineering, and Fluids Engineering Conference, Houston, 1983.